K.GOWTHAM-192372078

14. Construct a C program to organise the file using a single level directory.

Aim:

To construct a C program that organizes files using a single-level directory. The program will simulate basic file operations such as creating, displaying, and deleting files within the directory.

Algorithm:

1. **Create a Directory**: Simulate creating a directory to hold files.
2. **Add Files**: Simulate adding files to the directory.
3. **Display Files**: Display all the files currently in the directory.
4. **Delete Files**: Allow deletion of specific files from the directory.
5. **Search Files**: Allow the user to search for a specific file by name.

Procedure:

1. Define a structure for representing a file with its name and status (if it's in the directory).
2. Implement functions to create a file, delete a file, display all files, and search for a specific file.
3. Use an array to simulate the directory and store file information.
4. Implement a menu-driven interface to allow users to interact with the directory.

CODE:

```
#include <stdio.h>

#include <string.h>


#define MAX_FILES 10

#define MAX_FILE_NAME 50
```

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

File directory[MAX_FILES];

void initialize_directory() {
   for (int i = 0; i < MAX_FILES; i++) {
      directory[i].is_present = 0;
   }
}

int create_file(const char *file_name) {
   for (int i = 0; i < MAX_FILES; i++) {
      if (directory[i].is_present == 0) {
         strncpy(directory[i].file_name, file_name, MAX_FILE_NAME);
         directory[i].is_present = 1;
         return 1; // File created successfully
      }
   }
   return 0; // Directory full, file not created
}

int delete_file(const char *file_name) {
   for (int i = 0; i < MAX_FILES; i++) {
      if (directory[i].is_present == 1 && strcmp(directory[i].file_name, file_name) == 0) {
```

```
            directory[i].is_present = 0;
            return 1; // File deleted successfully
        }
    }
    return 0; // File not found
}

void display_files() {
    int found = 0;
    for (int i = 0; i < MAX_FILES; i++) {
        if (directory[i].is_present == 1) {
            printf("File: %s\n", directory[i].file_name);
            found = 1;
        }
    }
    if (!found) {
        printf("No files in the directory.\n");
    }
}

int search_file(const char *file_name) {
    for (int i = 0; i < MAX_FILES; i++) {
        if (directory[i].is_present == 1 && strcmp(directory[i].file_name, file_name) == 0) {
            return 1; // File found
        }
    }
    return 0; // File not found
}
```

```
int main() {
    int choice;
    char file_name[MAX_FILE_NAME];

    initialize_directory();

    while (1) {
        printf("\nMenu:\n");
        printf("1. Create a file\n");
        printf("2. Delete a file\n");
        printf("3. Display all files\n");
        printf("4. Search for a file\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // To consume the newline character after choice input

        switch (choice) {
            case 1:
                printf("Enter file name to create: ");
                fgets(file_name, MAX_FILE_NAME, stdin);
                file_name[strcspn(file_name, "\n")] = '\0';
                if (create_file(file_name)) {
                    printf("File '%s' created successfully.\n", file_name);
                } else {
                    printf("Directory is full. Cannot create file '%s'.\n", file_name);
                }
```

```
            break;
        case 2:
            printf("Enter file name to delete: ");
            fgets(file_name, MAX_FILE_NAME, stdin);
            file_name[strcspn(file_name, "\n")] = '\0';
            if (delete_file(file_name)) {
                printf("File '%s' deleted successfully.\n", file_name);
            } else {
                printf("File '%s' not found.\n", file_name);
            }
            break;
        case 3:
            printf("Displaying all files in the directory:\n");
            display_files();
            break;
        case 4:
            printf("Enter file name to search: ");
            fgets(file_name, MAX_FILE_NAME, stdin);
            file_name[strcspn(file_name, "\n")] = '\0';
            if (search_file(file_name)) {
                printf("File '%s' found in the directory.\n", file_name);
            } else {
                printf("File '%s' not found in the directory.\n", file_name);
            }
            break;
        case 5:
            printf("Exiting the program.\n");
            return 0;
```

```
        default:
            printf("Invalid choice. Please try again.\n");
      }
    }

    return 0;
}
```

OUTPUT: