K NARENDRA-192321162

1. 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:

