K NARENDRA-192321162

1. 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:

