Operating systems laboratory

Ex 12 – file organization techniques

Sreyas v

Cse c 185001162

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define MAX 100

#define MAX\_DIR 3

#define MAX\_FILE 3

typedef struct File

{

char name[25];

int start\_address;

} File;

void insertFileSingleLevel(File \*[]);

void displaySingleLevel(File \*[]);

typedef struct Directory

{

char name[25];

struct Directory \*subdir[MAX\_DIR];

File \*f[MAX\_FILE];

} Directory;

void init\_dir(Directory \*const);

void insertFileTree(Directory \*const);

void insertDirectoryTree(Directory \*const);

void displayTree(const Directory \*const, char path[]);

int main()

{

int choice, count = 0;

char name[30];

char path[100];

File \*arr[MAX], \*tmp = NULL;

for (int i = 0; i < MAX; i++)

arr[i] = NULL;

Directory root;

init\_dir(&root);

strcpy(root.name, "root");

while (1)

{

printf("\n\t\t\tFILE ORGANISATION TECHNIQUES\n");

printf(" 1 - Single Level Directory\n");

printf(" 2 - Tree Structure Directory\n");

printf(" 0 - Exit\n");

printf(" ---------------------------------\n");

printf(" Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 0:

exit(0);

case 1:

while (1)

{

printf("\n\n\t\tSINGLE LEVEL DIRECTORY\n");

printf(" 1 - Create a file\n");

printf(" 2 - List all files\n");

printf(" 0 - Back\n");

printf(" -----------------------------\n");

printf(" Enter your choice: ");

scanf("%d", &choice);

getchar();

if (choice == 0)

break;

switch (choice)

{

case 1:

insertFileSingleLevel(arr);

break;

case 2:

displaySingleLevel(arr);

break;

default:

printf(" Invalid Input!\n");

}

}

break;

case 2:

while (1)

{

printf("\n\n\t\tTREE STRUCTURE DIRECTORY\n");

printf(" 1 - Create a file\n");

printf(" 2 - Create a directory\n");

printf(" 3 - List all files\n");

printf(" 0 - Back\n");

printf(" -----------------------------\n");

printf(" Enter your choice: ");

scanf("%d", &choice);

getchar();

if (choice == 0)

break;

switch (choice)

{

case 1:

insertFileTree(&root);

break;

case 2:

insertDirectoryTree(&root);

break;

case 3:

strcpy(path, "/root");

printf(" +---------------------------+-------------------------------------+\n");

printf(" | File Name | Path |\n");

printf(" +---------------------------+-------------------------------------+\n");

displayTree(&root, path);

printf(" +---------------------------+-------------------------------------+\n");

break;

default:

printf(" Invalid Input!\n");

}

}

break;

default:

printf(" Invalid Input!\n");

break;

}

}

}

void init\_dir(Directory \*const dir)

{

strcpy(dir->name, "");

// for (int i = 0; i < 3; i++)

// dir->f[i] = dir->subdir[i] = NULL;

}

void insertFileSingleLevel(File \*root[])

{

File \*tmp = (File \*)malloc(sizeof(File));

printf(" Enter the name of the file: ");

scanf("%[^\n]", tmp->name);

tmp->start\_address = 500 \* (rand() % 20);

int found = 0;

for (int i = 0; i < MAX; i++)

if (root[i] == NULL)

{

root[i] = tmp;

break;

}

else if (strcmp(root[i]->name, tmp->name) == 0)

{

found = 1;

break;

}

if (found)

printf(" Duplicate file name!\n");

else

printf(" Successfully added file!\n");

}

void displaySingleLevel(File \*root[])

{

if (!root[0])

printf(" Empty Directory!\n");

else

{

printf(" +---------------------------+---------------+\n");

printf(" | File Name | Start Address |\n");

printf(" +---------------------------+---------------+\n");

for (int i = 0; i < MAX && root[i]; i++)

printf(" | %-25s | %-4d |\n", root[i]->name, root[i]->start\_address);

printf(" +---------------------------+---------------+\n");

}

}

void insertDirectoryTree(Directory \*const root)

{

char path[100];

printf(" Enter path to directory [root/.../...]: ");

scanf("%[^\n]", path);

char \*dir, \*new\_dir;

Directory \*cd = root;

int found = 0, created = 0;

dir = strtok(path, "/");

if (strcmp(path, "root"))

{

printf(" Path should start with root!\n");

return;

}

dir = strtok(NULL, "/");

if (!dir)

{

printf(" \nInvalid Directory Name!\n");

return;

}

while (dir != NULL)

{

for (int i = 0; i < MAX\_DIR; i++)

{

if (cd->subdir[i])

if (strcmp(dir, cd->subdir[i]->name) == 0)

{

cd = cd->subdir[i];

found = 1;

break;

}

}

new\_dir = dir;

dir = strtok(NULL, "/");

if (!found)

break;

}

if (dir == NULL)

{

for (int i = 0; i < MAX\_DIR; i++)

if (!cd->subdir[i])

{

cd->subdir[i] = (Directory \*)malloc(sizeof(Directory));

init\_dir(cd->subdir[i]);

strcpy(cd->subdir[i]->name, new\_dir);

created = 1;

break;

}

else if (strcmp(cd->subdir[i]->name, new\_dir) == 0)

break;

}

if (created)

printf(" Successfully created directory!\n");

else

printf(" Unable to create directory!\n");

}

void insertFileTree(Directory \*const root)

{

char path[100];

printf(" Enter path to files [root/.../...]: ");

scanf("%[^\n]", path);

char \*dir, \*new\_file;

Directory \*cd = root;

int found = 0, created = 0;

dir = strtok(path, "/");

if (strcmp(path, "root"))

{

printf(" Path should start with root!\n");

return;

}

dir = strtok(NULL, "/");

while (dir != NULL)

{

for (int i = 0; i < MAX\_DIR; i++)

{

if (cd->subdir[i])

if (strcmp(dir, cd->subdir[i]->name) == 0)

{

cd = cd->subdir[i];

found = 1;

break;

}

}

new\_file = dir;

dir = strtok(NULL, "/");

if (!found)

break;

}

if (dir == NULL)

{

for (int i = 0; i < MAX\_DIR; i++)

if (!cd->f[i])

{

cd->f[i] = (File \*)malloc(sizeof(File));

strcpy(cd->f[i]->name, new\_file);

created = 1;

break;

}

else if (strcmp(cd->f[i]->name, new\_file) == 0)

break;

}

if (created)

printf(" Successfully created File!\n");

else

printf(" Unable to create File!\n");

}

void displayTree(const Directory \*dir, char path[100])

{

for (int i = 0; i < MAX\_FILE; i++)

if (dir->f[i])

printf(" | %-25s | %-35s |\n", dir->f[i]->name, path);

for (int i = 0; i < MAX\_DIR; i++)

if (dir->subdir[i])

{

strcat(path, "/");

strcat(path, dir->subdir[i]->name);

displayTree(dir->subdir[i], path);

}

}

OUTPUT

C:\Users\sreyas\Desktop\labs\OS\File Organisation>gcc fileorganisation.c -o m

C:\Users\sreyas\Desktop\labs\OS\File Organisation>m

FILE ORGANISATION TECHNIQUES

1 - Single Level Directory

2 - Tree Structure Directory

0 - Exit

---------------------------------

Enter your choice: 1

SINGLE LEVEL DIRECTORY

1 - Create a file

2 - List all files

0 - Back

-----------------------------

Enter your choice: 1

Enter the name of the file: file1.txt

Successfully added file!

SINGLE LEVEL DIRECTORY

1 - Create a file

2 - List all files

0 - Back

-----------------------------

Enter your choice: 1

Enter the name of the file: ocd.pdf

Successfully added file!

SINGLE LEVEL DIRECTORY

1 - Create a file

2 - List all files

0 - Back

-----------------------------

Enter your choice: 2

+---------------------------+---------------+

| File Name | Start Address |

+---------------------------+---------------+

| file1.txt | 500 |

| ocd.pdf | 3500 |

+---------------------------+---------------+

SINGLE LEVEL DIRECTORY

1 - Create a file

2 - List all files

0 - Back

-----------------------------

Enter your choice: 1

Enter the name of the file: ocd.pdf

Duplicate file name!

SINGLE LEVEL DIRECTORY

1 - Create a file

2 - List all files

0 - Back

-----------------------------

Enter your choice: 0

FILE ORGANISATION TECHNIQUES

1 - Single Level Directory

2 - Tree Structure Directory

0 - Exit

---------------------------------

Enter your choice: 2

TREE STRUCTURE DIRECTORY

1 - Create a file

2 - Create a directory

3 - List all files

0 - Back

-----------------------------

Enter your choice: 1

Enter path to files [root/.../...]: root/file.txt