UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

Assignment 13 – File Orgnaisation Techniques: Single Level and Hierarchical Directory Structures

Date: 30/05/2022

Aim:

To develop a C program to implement the following file organization techniques:
a) Single level Directory
b) Hierarchical Structure

Algorithm:

```
1. Start
2. Let the user choose between single level and hierarchical directory structures.
3. Single Level Directory
    a. Maintain a table containing the filename and the starting address location of that file.
    b. Give options for creating a new file.
    c. Get the name of the file as input from the user. If the file does not already exist,
       increment the file counter and add the file to the directory.
    d. Update the table accordingly.
4. Tree Structured Directory
    a. Maintain tables for each directory starting from root.
    b. Create a structure for a node in tree which contains an array to hold directories and
       an array to hold files.
    c. Limit each directory to have a maximum of three sub-directories and files.
    d. For each sub-directory follow the same table structure as described above.
    e. Give options for creating a new directory or a new file.
    f. Get the name and path of the directory or file as input from the user.
    g. Update the table accordingly.
5. Stop
```

Code:

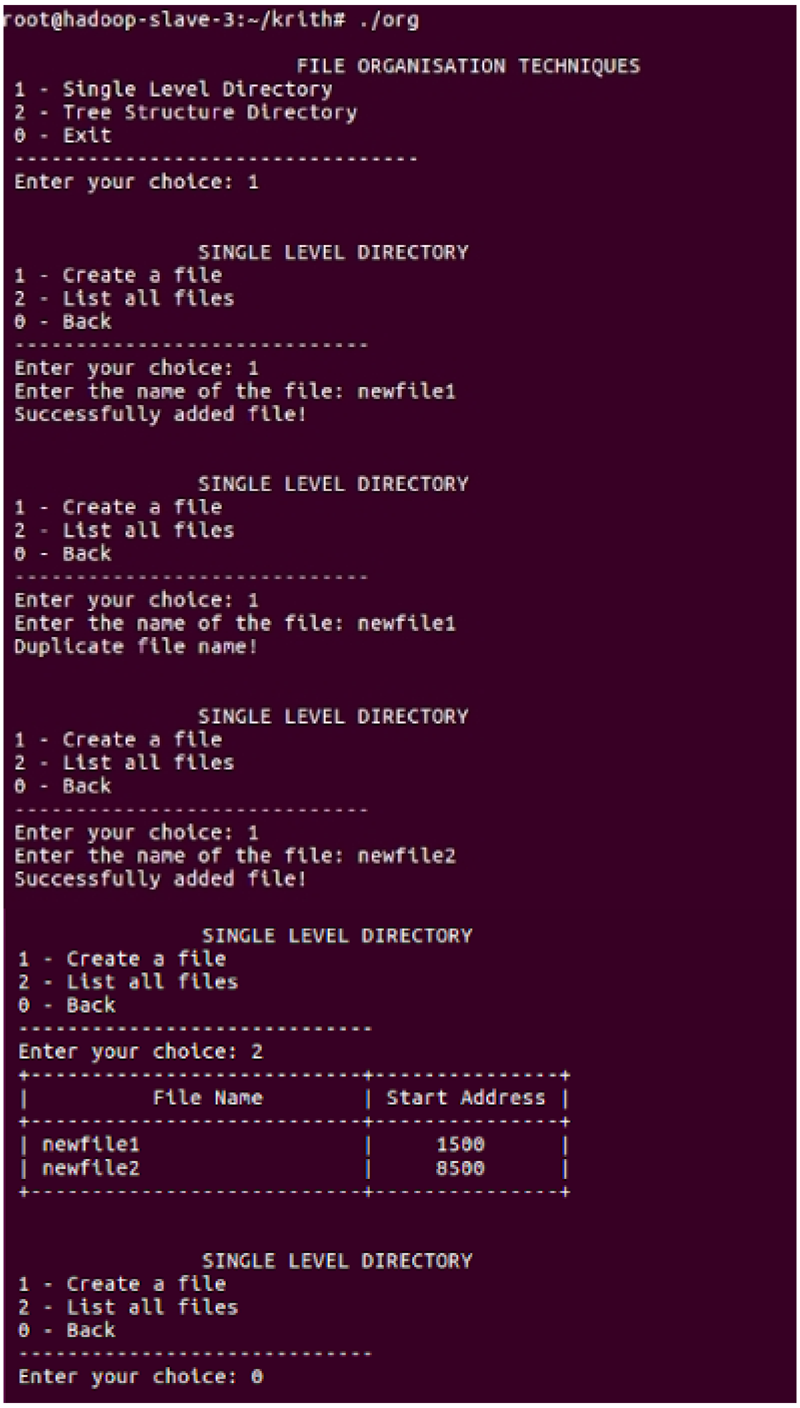
```
//Program to implement memory organisation techniques
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#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);
```

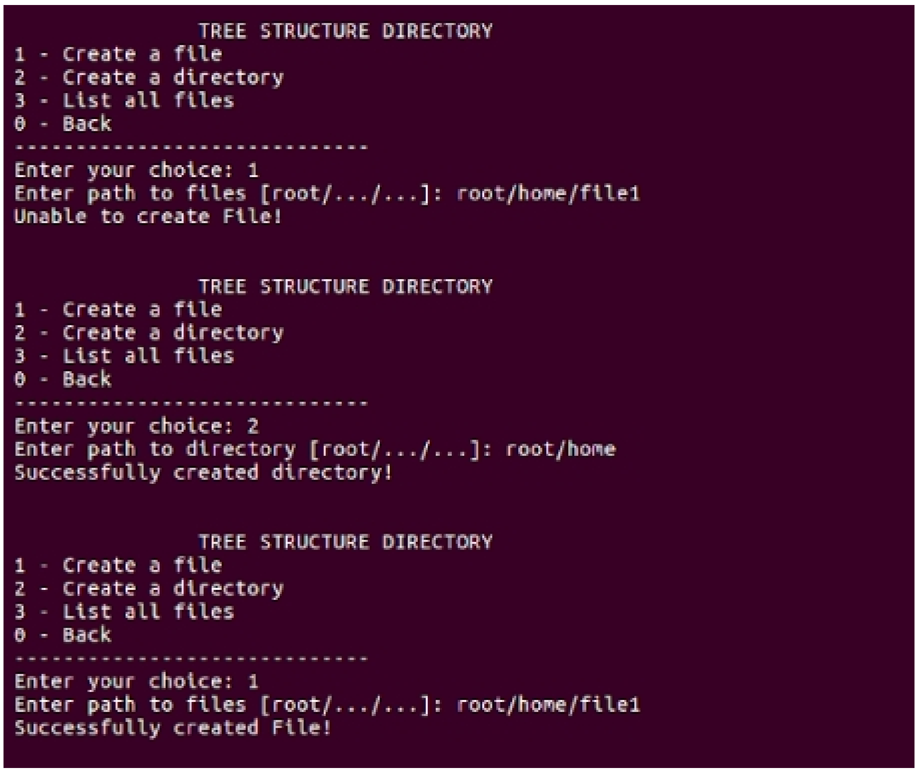
```
case 1:
    while (1)
    {
        printf("\n\n\t\tSINGLE LEVEL DIRECTORY\n");
        printf(" 1 - Create a file\n");
        printf(" 2 - List all files\n");
        printf(" 0 - Back\n");
        printf(" -----------------------------\n");
        printf(" Enter your choice: ");
        scanf("%d", &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 * (random() % 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);
        }
}
```

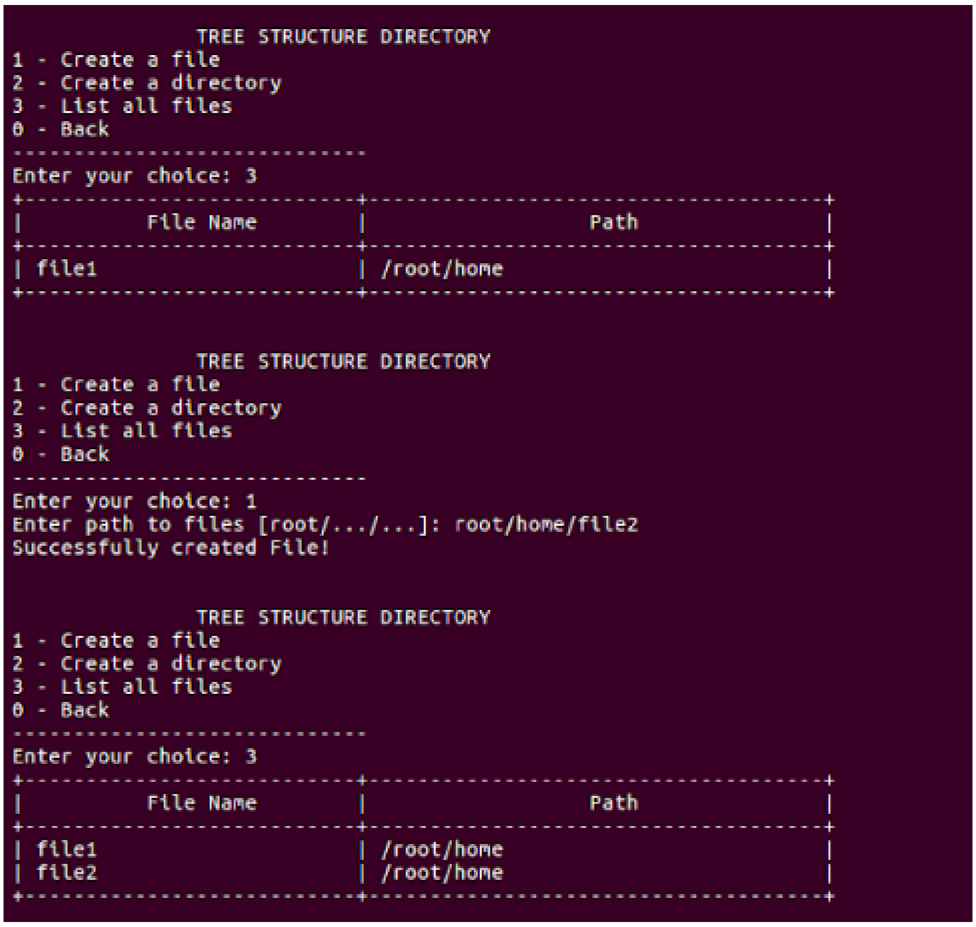
Output:



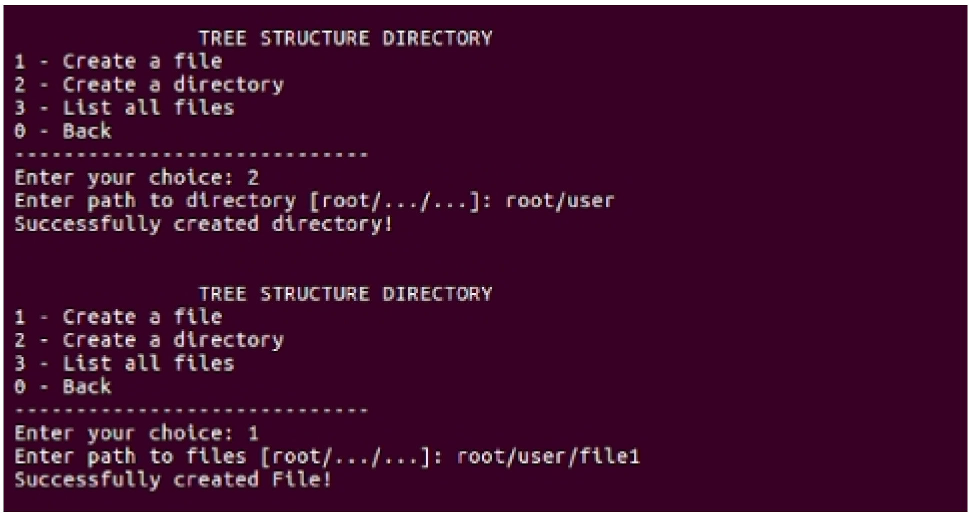
```
root@hadoop-slave-3:~/krith# ./org

                        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: newfile1
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: newfile1
Duplicate file name!


                SINGLE LEVEL DIRECTORY
1 - Create a file
2 - List all files
0 - Back
-----------------------------
Enter your choice: 1
Enter the name of the file: newfile2
Successfully added file!


                SINGLE LEVEL DIRECTORY
1 - Create a file
2 - List all files
0 - Back
-----------------------------
Enter your choice: 2
+---------------------------+---------------+
|          File Name        | Start Address |
+---------------------------+---------------+
| newfile1                  |     1500      |
| newfile2                  |     8500      |
+---------------------------+---------------+


                SINGLE LEVEL DIRECTORY
1 - Create a file
2 - List all files
0 - Back
-----------------------------
Enter your choice: 0
```



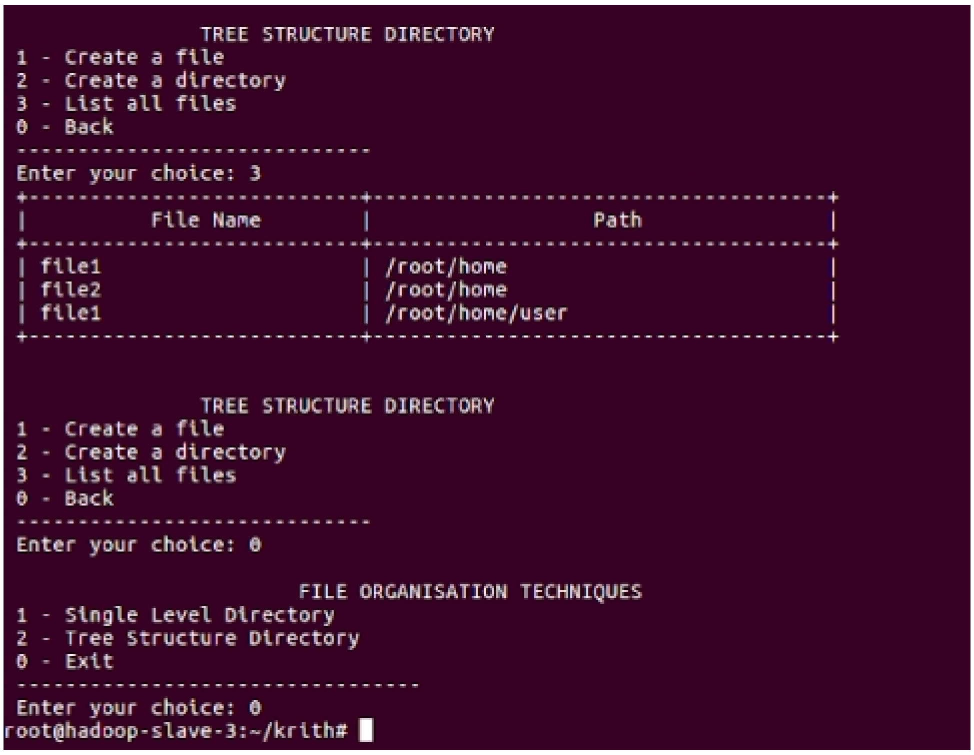
```
               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - List all files
0 - Back
-----------------------------
Enter your choice: 1
Enter path to files [root/.../...]: root/home/file1
Unable to create File!


               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - List all files
0 - Back
-----------------------------
Enter your choice: 2
Enter path to directory [root/.../...]: root/home
Successfully created directory!


               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/home/file1
Successfully created File!
```



```
               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - List all files
0 - Back
-----------------------------
Enter your choice: 3
+----------------------------+--------------------------------------+
|          File Name         |                 Path                 |
+----------------------------+--------------------------------------+
| file1                      | /root/home                           |
+----------------------------+--------------------------------------+


               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/home/file2
Successfully created File!


               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - List all files
0 - Back
-----------------------------
Enter your choice: 3
+----------------------------+--------------------------------------+
|          File Name         |                 Path                 |
+----------------------------+--------------------------------------+
| file1                      | /root/home                           |
| file2                      | /root/home                           |
+----------------------------+--------------------------------------+
```



```
               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - List all files
0 - Back
-----------------------------
Enter your choice: 2
Enter path to directory [root/.../...]: root/user
Successfully created directory!


               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/user/file1
Successfully created File!
```



```
               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - List all files
0 - Back
-----------------------------
Enter your choice: 3
+----------------------------+-------------------------------------+
|          File Name         |                Path                 |
+----------------------------+-------------------------------------+
| file1                      | /root/home                          |
| file2                      | /root/home                          |
| file1                      | /root/home/user                     |
+----------------------------+-------------------------------------+


               TREE STRUCTURE DIRECTORY
1 - Create a file
2 - Create a directory
3 - 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: 0
root@hadoop-slave-3:~/krith#
```

Name: Krithika Swaminathan
Roll No.: 205001057

Learning outcomes:

- File organisation techniques were understood and implemented.
- Single level and hierarchical level organisation was understood and implemented.
