EX:12 FILE ORGANIZATION TECHNIQUES

-S.Vishakan CSE-C 18 5001 196

SOURCE CODE – SINGLE LEVEL DIRECTORY:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{ //structure for file
   char name[50];
   int start_addr;
} file;

file *newFile(char name[], int addr);
int searchFile(file *root[], char name[], int file_count);
int insertFile(file *root[], int file_count);
void displayFiles(file *root[], int file_count);

int main(void)
{
   int opt = 1, file_count = 0;

   file *root[50];

   printf("\n\t\t\tSingle Level Directory Structure\n");
   while (opt != 0)
   {
      printf("\n\t\tMain Menu\n\t1. Insert a New File\n\t2. Display Existing Files\n\t0. Exit the
Program\n\tYour choice -> ");
      scanf("%d", &opt);
      switch (opt)
      {
      case 1:
         if (file_count < 50)
         {
            file_count += insertFile(root, file_count);
            printf("\nCan accomodate %d more files in this directory structure.\n", (50 - file_count));
         }
         break;

      case 2:
         displayFiles(root, file_count);
         break;
```

```
        case 0:
            printf("\n\t\tThank You!\n");
            break;

        default:
            printf("\n\tInvalid Option.\n");
            break;
        }
    }

    return 0;
}

file *newFile(char name[], int addr)
{
    file *new_file = (file *)malloc(sizeof(file));
    strcpy(new_file->name, name);
    new_file->start_addr = addr;

    return new_file;
}

int searchFile(file *root[], char name[], int file_count)
{
    int flag = 0, i = 0;

    for (i = 0; i < file_count; i++)
    {
        if (root[i] != NULL)
        {
            if (strcmp(root[i]->name, name) == 0)
            {
                flag = 1;
                break;
            }
        }
    }

    return flag;
}

int insertFile(file *root[], int file_count)
{
    char name[50];
    int flag = 0, addr = 0;

    printf("\nEnter the File Name: ");
    scanf("%s", name);
    printf("\nEnter the Starting Address of File %s: ", name);
    scanf("%d", &addr);
    if (searchFile(root, name, file_count) == 1)
    {
        printf("\nFile %s already exists!\n", name);
        flag = 0;
    }

    else
    {
        root[file_count] = newFile(name, addr);
        printf("\nCreated File %s.\n", name);
        flag = 1;
    }

    return flag;
}

void displayFiles(file *root[], int file_count)
{
    printf("\nContents of Root Directory: \n");

    if (file_count == 0)
    {
        printf("\nRoot directory is empty.\n");
    }

    else
    {
        int i = 0;

        printf("\n----------------------------------------------");
        printf("\n|\tFile\t|\tStarting Address\t|");
        printf("\n----------------------------------------------\n");

        for (i = 0; i < file_count; i++)
        {
            if (root[i] != NULL)
            {
                printf("|\t%s\t|\t\t%d\t\t|\n", root[i]->name, root[i]->start_addr);
            }
        }
        printf("----------------------------------------------\n");
    }
}
```

OUTPUT – SINGLE LEVEL DIRECTORY:

```
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ gcc
SingleLevel.c -o s
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ ./s

                    Single Level Directory Structure

             Main Menu
      1. Insert a New File
      2. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter the File Name: OS

Enter the Starting Address of File OS: 1344

Created File OS.

Can accomodate 49 more files in this directory structure.

             Main Menu
      1. Insert a New File
      2. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter the File Name: DBMS

Enter the Starting Address of File DBMS: 1750

Created File DBMS.

Can accomodate 48 more files in this directory structure.

             Main Menu
      1. Insert a New File
      2. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter the File Name: DAA

Enter the Starting Address of File DAA: 2190

Created File DAA.

Can accomodate 47 more files in this directory structure.
```

```
          Main Menu
      1. Insert a New File
      2. Display Existing Files
      0. Exit the Program
      Your choice -> 2

Contents of Root Directory:

-------------------------------------------------
|     File    |      Starting Address       |
-------------------------------------------------
|     OS      |            1344            |
|     DBMS    |            1750            |
|     DAA     |            2190            |
-------------------------------------------------

          Main Menu
      1. Insert a New File
      2. Display Existing Files
      0. Exit the Program
      Your choice -> 0

          Thank You!
```

SOURCE CODE – TWO LEVEL DIRECTORY:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{ //file structure
   char name[50];
} file;

typedef struct
{ //directory structure with capacity of 5 files
   char name[50];
   int capacity;
   file *list[5];
} directory;

typedef struct
{   //general structure to represent file/directory
   int type;   //type -> 0 for FILE, type -> 1 for DIRECTORY
   void *pointer;
} unit;

unit root[50];  //array to hold 50 units of files/directories
int count = 0;

file *newFile(char name[]);
directory *newDirectory(char name[]);
int searchFile(char name[]);
directory *searchDirectory(char name[]);
void insertUnit(char name[], int type);
void insertFiletoDir(directory *dir, char name[]);
void displayContents(unit root[]);

int main(void)
{
   int opt = 1, type = 0;
   char name[50], dir_name[50];

   printf("\n\t\t\tTwo Level Directory Structure\n");
   while (opt != 0)
   {
      printf("\n\t\tMain Menu\n\t1. Create a New File\n\t2. Create a New Directory\n\t3. Display
Existing Files\n\t0. Exit the Program\n\tYour choice -> ");
      scanf("%d", &opt);
      switch (opt)
      {
      case 1:
         printf("\nEnter \"root\" to create a file in the root directory.\nEnter \"root/directory\" to
create a file in the sub-directory.");
```

```
            printf("\nEnter the Directory Name: ");
            scanf("%s", dir_name);
            printf("\nEnter the File Name: ");
            scanf("%s", name);
            if (strcmp(dir_name, "root") != 0)
            {   //if user enters a sub-directory
                char *sub_dir = strtok(dir_name, "/");  //split to find sub-dir from input
                sub_dir = strtok(NULL, "/");
                directory *dir = searchDirectory(sub_dir);
                if (dir != NULL)
                {   //inserting file to the sub-directory
                    insertFiletoDir(dir, name);
                }
                else
                {   //invalid sub-directory entered by user
                    printf("\nDirectory %s does not exist.", dir_name);
                }
            }

            else if (strcmp(dir_name, "root") == 0)
            {   //if user enters root as directory
                insertUnit(name, 0);    //file
            }

            printf("\nCan accomodate %d more files in this directory structure.\n", (50 - count));
            break;

        case 2:
            printf("\nEnter the Directory Name: ");
            scanf("%s", dir_name);
            insertUnit(dir_name, 1);    //directory
            break;

        case 3:
            displayContents(root);
            break;

        case 0:
            printf("\n\t\tThank You!\n");
            break;

        default:
            printf("\n\tInvalid Option.\n");
            break;
        }
    }

    return 0;
}
```

```
file *newFile(char name[])
{   //making a new file structure
    file *new_file = (file *)malloc(sizeof(file));
    strcpy(new_file->name, name);

    return new_file;
}

directory *newDirectory(char name[])
{   //making a new directory structure
    int i = 0;

    directory *new_dir = (directory *)malloc(sizeof(directory));
    strcpy(new_dir->name, name);
    new_dir->capacity = 0;

    for (i = 0; i < 5; i++)
    {   //initialise
        new_dir->list[i] = NULL;
    }

    return new_dir;
}

int searchFile(char name[])
{   //searching a file under root directory
    int flag = 0, i = 0;

    for (i = 0; i < count; i++)
    {
        if (root[i].pointer != NULL)
        {
            if (strcmp(((file *)(root[i].pointer))->name, name) == 0)
            {
                flag = 1;
                break;
            }
        }
    }

    return flag;
}

directory *searchDirectory(char name[])
{   //searching for a directory under root directory
    directory *flag_dir = NULL;
    int i = 0;

    for (i = 0; i < count; i++)
    {
        if (root[i].pointer != NULL)
        {
            if (strcmp(((directory *)(root[i].pointer))->name, name) == 0)
            {
                flag_dir = ((directory *)(root[i].pointer));
                break;
            }
        }
    }

    return flag_dir;    //pointer to desired directory
}

void insertUnit(char name[], int type)
{   //inserting a new file or directory under root
    if (count >= 50)
    {   //capacity reached
        printf("\nReached maximum capacity.\n");
        return;
    }

    if (type == 0 && searchFile(name) == 1)
    {   //if file and it already exists
        printf("\nFile %s already exists!\n", name);
        return;
    }

    if (type == 1 && searchDirectory(name) != NULL)
    {   //if directory and it already exists
        printf("\nDirectory %s already exists!\n", name);
        return;
    }

    if (type == 1 && searchFile(name) == 1)
    {   //if directory and already a file with the same name exists
        printf("\nFile named %s already exists!\n", name);
        return;
    }

    if (type == 0)
    {   //creating the file
        root[count].pointer = newFile(name);
        printf("\nCreated File %s.\n", name);
    }
    else
    {   //creating the directory
        root[count].pointer = newDirectory(name);
        printf("\nCreated Directory %s.\n", name);
    }

    root[count].type = type;
    count++;
}
```

```
void insertFiletoDir(directory *dir, char name[])
{   //inserting the file to directory
    int i, pos;
    if (dir->capacity >= 5)
    {
        printf("\nDirectory is Full!\n");
    }
    else
    {
        for (i = 0; i < 5; i++)
        {
            if (dir->list[i] != NULL)
            {   //moving through existing files
                if (strcmp(dir->list[i]->name, name) == 0)
                {   //if file already exists
                    printf("\nFile %s already exists!\n", name);
                    return;
                }
            }
            else
            {   //found the position, breaking
                pos = i;
                break;
            }
        }
        dir->list[pos] = newFile(name);     //inserting
        dir->capacity += 1;
        printf("\nCreated File %s in Directory %s.\n", name, dir->name);
    }
}

void displayContents(unit root[])
{   //display the entire contents under root directory
    printf("\nContents of Root Directory:\n");
    if (count == 0)
    {   //if empty
        printf("\nRoot directory is empty\n");
        return;
    }
    else
    {
        int dir_count = 0, file_count = 0, i = 0, j = 0;
        printf("\nFiles:\n");   //all files under root

        for (int i = 0; i < count; i++)
        {
            if (root[i].pointer != NULL)
            {
                if (root[i].type == 0)
                {
                    file_count++;
                    printf("%s ", ((file *)(root[i].pointer))->name);
                }
            }
        }

        if (file_count == 0)
        {
            printf("\t\t-NIL-");
        }

        printf("\n");
        file_count = 0;
        printf("\nDirectories:\n");     //all directories under root

        for (i = 0; i < count; i++)
        {
            if (root[i].pointer != NULL)
            {
                if (root[i].type == 1)
                {
                    dir_count++;
                    printf("%s ", ((directory *)(root[i].pointer))->name);
                }
            }
        }

        if (dir_count == 0)
        {
            printf("\t\t-NIL-");
        }

        printf("\n");
        dir_count = 0;

        for (i = 0; i < count; i++)     //contents of directories under root
        {
            if (root[i].pointer != NULL)
            {
                if (root[i].type == 1)
                {
                    dir_count++;
                    printf("\nContents of Directory %s:\n", ((directory *)(root[i].pointer))->name);
                    file_count = 0;
                    for (j = 0; j < 5; j++)
                    {   //traversing the directory list of files
                        if (((directory *)(root[i].pointer))->list[j] != NULL)
                        {
                            printf("%s ", ((directory *)(root[i].pointer))->list[j]->name);
                            file_count++;
                        }
                    }
                    if (file_count == 0)
                    {
```

```
                        printf("\t\t-NIL-");
                    }
                    printf("\n");
                }
            }
        }

        printf("\n");
    }
}
```

OUTPUT – TWO LEVEL DIRECTORY:

```
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ gcc TwoLevel.c -o t
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ ./t

                    Two Level Directory Structure

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 2

Enter the Directory Name: OS

Created Directory OS.

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 2

Enter the Directory Name: DBMS

Created Directory DBMS.

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.
Enter the Directory Name: root/OS

Enter the File Name: Semaphores

Created File Semaphores in Directory OS.

Can accomodate 48 more files in this directory structure.

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
```

```
        Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.
Enter the Directory Name: root/OS

Enter the File Name: Paging

Created File Paging in Directory OS.

Can accomodate 48 more files in this directory structure.

                Main Menu
        1. Create a New File
        2. Create a New Directory
        3. Display Existing Files
        0. Exit the Program
        Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.
Enter the Directory Name: root

Enter the File Name: N_Queens.py

Created File N_Queens.py.

Can accomodate 47 more files in this directory structure.


                Main Menu
        1. Create a New File
        2. Create a New Directory
        3. Display Existing Files
        0. Exit the Program
        Your choice -> 3

Contents of Root Directory:

Files:
N_Queens.py

Directories:
OS DBMS

Contents of Directory OS:
Semaphores Paging

Contents of Directory DBMS:
                -NIL-
        Main Menu
1. Create a New File
2. Create a New Directory
3. Display Existing Files
0. Exit the Program
Your choice -> 0

        Thank You!
```

SOURCE CODE – HIERARCHICAL STRUCTURE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{   //struct for file
    char name[50];
} file;

typedef struct Dir
{   //struct for tree-like directory
    char name[50];
    struct Dir *dir1, *dir2, *dir3;
    file *file1, *file2;
} dir;

dir *root = NULL;   //root directory

dir *initRoot();
void insertDirectory(char path[]);
void insertFile(char path[]);
void displayContents(dir *root, char path[]);

int main(void)
{
    root = initRoot();
    int opt = 1;
    char dir_name[50], name[50], path[500];

    printf("\n\t\t\tTwo Level Directory Structure\n");

    while (opt != 0)
    {
        printf("\n\t\tMain Menu\n\t1. Create a New File\n\t2. Create a New Directory\n\t3. Display
Existing Files\n\t0. Exit the Program\n\tYour choice -> ");
        scanf("%d", &opt);
        switch (opt)
        {
        case 1:
            printf("\nEnter \"root\" to create a file in the root directory.\nEnter \"root/directory\" to
create a file in the sub-directory.");
            printf("\n\nEnter the Path: ");
            scanf("%s", dir_name);
            insertFile(dir_name);
            break;

        case 2:
            printf("\nEnter \"root\" to create a file in the root directory.\nEnter \"root/directory\" to
create a file in the sub-directory.");
```

```
            printf("\n\nEnter the Path: ");
            scanf("%s", dir_name);
            insertDirectory(dir_name);
            break;

        case 3:
            strcpy(path, "");
            printf("\nFile\t\t\t\tPath\n\n");
            displayContents(root, path);
            break;

        case 0:
            printf("\n\t\tThank You!\n");
            break;

        default:
            printf("\n\tInvalid Option.\n");
            break;
        }
    }
    return 0;
}

dir *initRoot()
{   //initialising root directory
    root = (dir *)malloc(sizeof(dir));

    strcpy(root->name, "root");
    root->dir1 = NULL;
    root->dir2 = NULL;
    root->dir3 = NULL;
    root->file1 = NULL;
    root->file2 = NULL;

    return root;
}

void insertDirectory(char path[])
{   //inserting a new directory to specified path
    dir *temp = root;
    char *dir_name = strtok(path, "/");
    dir_name = strtok(NULL, "/");

    while (dir_name != NULL)
    {   //moving to the specified sub-directory
        if (temp->dir1 != NULL && strcmp(dir_name, temp->dir1->name) == 0)
        {
            temp = temp->dir1;
        }
        else if (temp->dir2 != NULL && strcmp(dir_name, temp->dir2->name) == 0)
        {
            temp = temp->dir2;
        }
        else if (temp->dir3 != NULL && strcmp(dir_name, temp->dir3->name) == 0)
        {
            temp = temp->dir3;
        }

        dir_name = strtok(NULL, "/");
    }

    if (dir_name == NULL)
    {
        if (temp->dir1 == NULL || temp->dir2 == NULL || temp->dir3 == NULL)
        {   //if space exists in the specified sub-directory
            char dirname[50];
            printf("\nEnter the Directory Name: ");
            scanf("%s", dirname);

            dir *new_dir = (dir *)malloc(sizeof(dir));
            new_dir->dir1 = NULL;
            new_dir->dir2 = NULL;
            new_dir->dir3 = NULL;
            new_dir->file1 = NULL;
            new_dir->file2 = NULL;
            strcpy(new_dir->name, dirname);
            //connecting it to a free pointer of the parent directory
            if (temp->dir1 == NULL)
            {
                temp->dir1 = new_dir;
            }
            else if (temp->dir2 == NULL && strcmp(dirname, temp->dir1->name) != 0)
            {
                temp->dir2 = new_dir;
            }
            else if (strcmp(dirname, temp->dir1->name) != 0 && strcmp(dirname, temp->dir2->name) !
= 0)
            {
                temp->dir3 = new_dir;
            }
            else if (strcmp(dirname, temp->dir1->name) == 0 || strcmp(dirname, temp->dir2->name) ==
0)
            {   //if it already exists
                printf("\nDirectory %s already exists!\n", dirname);
            }
            else
            {   //if no space is free in the sub-directory
                printf("\nDirectory Limit Exceeded.(Only 3 directories allowed under any directory)\n");
            }
        }
    }
}
```

```
void insertFile(char path[])
{   //inserting a new file to specified path
    dir *temp = root;
    char *split = strtok(path, "/");
    split = strtok(NULL, "/");

    while (split != NULL)
    {   //moving to specified sub-directory
        if (temp->dir1 != NULL && strcmp(split, temp->dir1->name) == 0)
        {
            temp = temp->dir1;
        }
        else if (temp->dir2 != NULL && strcmp(split, temp->dir2->name) == 0)
        {
            temp = temp->dir2;
        }
        else if (temp->dir3 != NULL && strcmp(split, temp->dir3->name) == 0)
        {
            temp = temp->dir3;
        }
        split = strtok(NULL, "/");
    }

    if (split == NULL)
    {
        if (temp->file1 == NULL || temp->file2 == NULL)
        {   //if the sub-directory has space for files
            char file_name[50];
            printf("\nEnter the File Name: ");
            scanf("%s", file_name);

            file *new_file = (file *)malloc(sizeof(file));
            strcpy(new_file->name, file_name);

            if (temp->file1 == NULL)
            {
                temp->file1 = new_file;
            }
            else if (temp->file2 == NULL)
            {
                temp->file2 = new_file;
            }
        }
        else
        {   //if it doesn't have space for files
            printf("\nFile Limit Exceeded.(Only 2 Files allowed in any directory)\n");
        }
    }
}
```

```
void displayContents(dir *root, char path[])
{   //to display the contents of the a directory
    char temp[50];

    if (root != NULL)
    {
        strcat(path, root->name);
        strcat(path, "/");
        if (root->file1 != NULL)
        {
            printf("%s\t\t\t%s\n", root->file1->name, path);
        }
        if (root->file2 != NULL)
        {
            printf("%s\t\t\t%s\n", root->file2->name, path);
        }
        if (root->dir1 != NULL)
        {
            strcpy(temp, path);
            displayContents(root->dir1, temp);
        }
        if (root->dir2 != NULL)
        {
            strcpy(temp, path);
            displayContents(root->dir2, temp);
        }
        if (root->dir3 != NULL)
        {
            strcpy(temp, path);
            displayContents(root->dir3, temp);
        }
    }
}
```

OUTPUT – HIERARCHICAL STRUCTURE:

```
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ gcc
TreeHierarchy.c -o t
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ ./t

                    Two Level Directory Structure

             Main Menu
        1. Create a New File
        2. Create a New Directory
        3. Display Existing Files
        0. Exit the Program
        Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root

Enter the File Name: N_Queens.py

             Main Menu
        1. Create a New File
        2. Create a New Directory
        3. Display Existing Files
        0. Exit the Program
        Your choice -> 2

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root

Enter the Directory Name: OS

             Main Menu
        1. Create a New File
        2. Create a New Directory
        3. Display Existing Files
        0. Exit the Program
        Your choice -> 2

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root/OS

Enter the Directory Name: Paging
```

```
          Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root/OS/Paging

Enter the File Name: Paging.c

          Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root/OS

Enter the File Name: Queue.h

          Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 3

File          Path

N_Queens.py     root/
Queue.h         root/OS/
Paging.c        root/OS/Paging/

          Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Display Existing Files
      0. Exit the Program
      Your choice -> 0

          Thank You!
```

SOURCE CODE – DIRECTED ACYCLIC GRAPH:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{ //struct for file
    char name[50];
} file;

typedef struct Dir
{ //struct for tree-like directory
    char name[50];
    struct Dir *dir1, *dir2, *dir3;
    file *file1, *file2;
} dir;

dir *root = NULL; //root directory

dir *initRoot();
void insertDirectory(char path[]);
void insertFile(char path[]);
void displayContents(dir *root, char path[]);
file *getFilePointer(char path[]);
dir *getDirectoryPointer(char path[]);
void createLink(char path[], char dir_name[]);

int main(void)
{
    root = initRoot();
    int opt = 1;
    char dir_name[50], name[50], path[500], path_name[50];

    printf("\n\t\t\tTwo Level Directory Structure\n");

    while (opt != 0)
    {
        printf("\n\t\tMain Menu\n\t1. Create a New File\n\t2. Create a New Directory\n\t3. Create a
Link to a File\n\t4. Display Existing Files\n\t0. Exit the Program\n\tYour choice -> ");
        scanf("%d", &opt);
        switch (opt)
        {
        case 1:
            printf("\nEnter \"root\" to create a file in the root directory.\nEnter \"root/directory\" to
create a file in the sub-directory.");
            printf("\n\nEnter the Path: ");
            scanf("%s", dir_name);
            insertFile(dir_name);
            break;
        case 2:
            printf("\nEnter \"root\" to create a file in the root directory.\nEnter \"root/directory\" to
create a file in the sub-directory.");
            printf("\n\nEnter the Path: ");
            scanf("%s", dir_name);
            insertDirectory(dir_name);
            break;

        case 3:
            printf("\nEnter the Path of File (Including File Name): ");
            scanf("%s", path_name);
            printf("\nEnter the Path of Directory to Create Link in: ");
            scanf("%s", dir_name);
            createLink(path_name, dir_name);
            break;

        case 4:
            strcpy(path, "");
            printf("\nFile\t\t\t\tPath\n\n");
            displayContents(root, path);
            break;

        case 0:
            printf("\n\t\tThank You!\n");
            break;

        default:
            printf("\n\tInvalid Option.\n");
            break;
        }
    }
    return 0;
}

dir *initRoot()
{ //initialising root directory
    root = (dir *)malloc(sizeof(dir));

    strcpy(root->name, "root");
    root->dir1 = NULL;
    root->dir2 = NULL;
    root->dir3 = NULL;
    root->file1 = NULL;
    root->file2 = NULL;

    return root;
}
```

```
void insertDirectory(char path[])
{ //inserting a new directory to specified path
    dir *temp = root;
    char *dir_name = strtok(path, "/");
    dir_name = strtok(NULL, "/");

    while (dir_name != NULL)
    { //moving to the specified sub-directory
        if (temp->dir1 != NULL && strcmp(dir_name, temp->dir1->name) == 0)
        {
            temp = temp->dir1;
        }
        else if (temp->dir2 != NULL && strcmp(dir_name, temp->dir2->name) == 0)
        {
            temp = temp->dir2;
        }
        else if (temp->dir3 != NULL && strcmp(dir_name, temp->dir3->name) == 0)
        {
            temp = temp->dir3;
        }

        dir_name = strtok(NULL, "/");
    }

    if (dir_name == NULL)
    {
        if (temp->dir1 == NULL || temp->dir2 == NULL || temp->dir3 == NULL)
        { //if space exists in the specified sub-directory
            char dirname[50];
            printf("\nEnter the Directory Name: ");
            scanf("%s", dirname);

            dir *new_dir = (dir *)malloc(sizeof(dir));
            new_dir->dir1 = NULL;
            new_dir->dir2 = NULL;
            new_dir->dir3 = NULL;
            new_dir->file1 = NULL;
            new_dir->file2 = NULL;
            strcpy(new_dir->name, dirname);
            //connecting it to a free pointer of the parent directory
            if (temp->dir1 == NULL)
            {
                temp->dir1 = new_dir;
            }
            else if (temp->dir2 == NULL && strcmp(dirname, temp->dir1->name) != 0)
            {
                temp->dir2 = new_dir;
            }
            else if (strcmp(dirname, temp->dir1->name) != 0 && strcmp(dirname, temp->dir2->name) !
= 0)
            {
                temp->dir3 = new_dir;
            }
            else if (strcmp(dirname, temp->dir1->name) == 0 || strcmp(dirname, temp->dir2->name) ==
0)
            { //if it already exists
                printf("\nDirectory %s already exists!\n", dirname);
            }
            else
            { //if no space is free in the sub-directory
                printf("\nDirectory Limit Exceeded.(Only 3 directories allowed under any directory)\n");
            }
        }
    }
}

void insertFile(char path[])
{ //inserting a new file to specified path
    dir *temp = root;
    char *split = strtok(path, "/");
    split = strtok(NULL, "/");

    while (split != NULL)
    { //moving to specified sub-directory
        if (temp->dir1 != NULL && strcmp(split, temp->dir1->name) == 0)
        {
            temp = temp->dir1;
        }
        else if (temp->dir2 != NULL && strcmp(split, temp->dir2->name) == 0)
        {
            temp = temp->dir2;
        }
        else if (temp->dir3 != NULL && strcmp(split, temp->dir3->name) == 0)
        {
            temp = temp->dir3;
        }
        split = strtok(NULL, "/");
    }

    if (split == NULL)
    {
        if (temp->file1 == NULL || temp->file2 == NULL)
        { //if the sub-directory has space for files
            char file_name[50];
            printf("\nEnter the File Name: ");
            scanf("%s", file_name);

            file *new_file = (file *)malloc(sizeof(file));
            strcpy(new_file->name, file_name);

            if (temp->file1 == NULL)
            {
                temp->file1 = new_file;
            }
            else if (temp->file2 == NULL)
            {
                temp->file2 = new_file;
            }
        }
        else
        { //if it doesn't have space for files
            printf("\nFile Limit Exceeded.(Only 2 Files allowed in any directory)\n");
        }
    }
}

void displayContents(dir *root, char path[])
{ //to display the contents of the a directory
    char temp[50];

    if (root != NULL)
    {
        strcat(path, root->name);
        strcat(path, "/");
        if (root->file1 != NULL)
        {
            printf("%s\t\t\t%s\n", root->file1->name, path);
        }
        if (root->file2 != NULL)
        {
            printf("%s\t\t\t%s\n", root->file2->name, path);
        }
        if (root->dir1 != NULL)
        {
            strcpy(temp, path);
            displayContents(root->dir1, temp);
        }
        if (root->dir2 != NULL)
        {
            strcpy(temp, path);
            displayContents(root->dir2, temp);
        }
        if (root->dir3 != NULL)
        {
            strcpy(temp, path);
            displayContents(root->dir3, temp);
        }
    }
}
```

```
file *getFilePointer(char path[])
{ //to return the file pointer to the specified file
   dir *temp = root;
   char *split = strtok(path, "/");
   char *t;

   while (split != NULL)
   {   //traversing to the specified sub-directory
      if (temp->dir1 != NULL && strcmp(split, temp->dir1->name) == 0)
      {
         temp = temp->dir1;
      }
      else if (temp->dir2 != NULL && strcmp(split, temp->dir2->name) == 0)
      {
         temp = temp->dir2;
      }
      else if (temp->dir3 != NULL && strcmp(split, temp->dir3->name) == 0)
      {
         temp = temp->dir3;
      }
      t = split;
      split = strtok(NULL, "/");
      if (split == NULL)
      {   //reached the parent directory of the file
         if (strcmp(temp->file1->name, t) == 0)
         {
            return temp->file1;
         }
         else if (strcmp(temp->file2->name, t) == 0)
         {
            return temp->file2;
         }
         else
         {
            printf("\nThe specified file does not exist.\n");
            return NULL;
         }
      }
   }
   return NULL;
}

dir *getDirectoryPointer(char path[])
{ //to return the directory pointer to the specified directory
   char *split = strtok(path, "/");
   dir *temp = root;

   while (split != NULL)
   {   //traversing to the specified sub-directory
      if (temp->dir1 != NULL && strcmp(split, temp->dir1->name) == 0)
      {
         temp = temp->dir1;
```

```
        }
        else if (temp->dir2 != NULL && strcmp(split, temp->dir2->name) == 0)
        {
            temp = temp->dir2;
        }
        else if (temp->dir3 != NULL && strcmp(split, temp->dir3->name) == 0)
        {
            temp = temp->dir3;
        }

        split = strtok(NULL, "/");

        if (split == NULL)
        {   //reached the required directory
            return temp;
        }
    }
    return NULL;
}

void createLink(char path[], char dir_name[])
{   //creating a link to existing file to another directory
    file *temp_file = getFilePointer(path);
    dir *temp_dir = getDirectoryPointer(dir_name);

    if (temp_file != NULL)
    {
        if (temp_dir->file1 == NULL)
        {
            temp_dir->file1 = temp_file;
        }
        else if (temp_dir->file2 == NULL)
        {
            temp_dir->file2 = temp_file;
        }
        else
        {
            printf("\nThe destination directory is full. Link cannot be created.\n");
        }
    }
}
```

OUTPUT – DIRECTED ACYCLIC GRAPH:

```
(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization Techniques$ gcc
DAGraph.c -o d(base) vishakan@Legion:~/Desktop/Operating-Systems/Ex 12 File Organization
Techniques$ ./d

                    Two Level Directory Structure

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Create a Link to a File
      4. Display Existing Files
      0. Exit the Program
      Your choice -> 1

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root

Enter the File Name: DAG.c

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Create a Link to a File
      4. Display Existing Files
      0. Exit the Program
      Your choice -> 2

Enter "root" to create a file in the root directory.
Enter "root/directory" to create a file in the sub-directory.

Enter the Path: root

Enter the Directory Name: OS

             Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Create a Link to a File
      4. Display Existing Files
      0. Exit the Program
      Your choice -> 3

Enter the Path of File (Including File Name): root/DAG.c

Enter the Path of Directory to Create Link in: root/OS
```

```
          Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Create a Link to a File
      4. Display Existing Files
      0. Exit the Program
      Your choice -> 4

File                    Path

DAG.c                 root/
DAG.c                 root/OS/

          Main Menu
      1. Create a New File
      2. Create a New Directory
      3. Create a Link to a File
      4. Display Existing Files
      0. Exit the Program
      Your choice -> 0

          Thank You!
```