Department of Computer Engineering

BLG413E – SYSTEM PROGRAMMING
Project-3 Report

Group Number : G37

Group Members :

ID	Name	Surname
150160537	Okan	YILDIRIM
150160530	Özgür	AKTAŞ

Instructor : Lec. H. Turgut UYAR

1. Introduction

Main aim of the this project is implementing file system by using FUSE in Linux Operating System. With this project we have a chance to develop file system in Userspace by using the information we learned in class and compare cons and pros of developing in user-space with the previous projects. We experience that studying on user-space is least risky and fastest among the compiling kernel space and using kernel modules even though its performance is worst. In addition, it has advantage to use external libraries.

In this project, we implemented a file system which is constructed according to the a CSV file which contains Turkish postal codes data. In this file, there is six column: CODE, NEIGHBIRHOOD, CITY, DISTRICT, LATITUDE and LONGTITUDE. It is requested that there will be two directories in the top level directory: NAMES and CODES. In Names directory, there will be all city subdirectories. Each city directories will show the its districts as subdirectories and finally each district directory will show its neighborhood as text file. This text file contain data in a form of key-value pairs for each column. On the other hand, CODES directory will display every city code which is the two digits of the postal codes. Each city codes subdirectory will show the its neighborhood code as text file. Moreover, this FUSE-based file system will allow to delete and rename operations for only neighborhood files. These modifications will update the CSV file also. In same manner, any changes on the CSV file will be update file system accordingly. Consequently, general description of the project is given above.

2. Method and Detailed Explanation of Codes

Before we have been writing the project, we had checked if the FUSE was installed or not on to our Linux Machine. We saw that its 26th version was installed. Then, we implemented the general architecture of the project. Firstly, we defined the FUSE version and included required libraries. This part of code is given below:

```
#define FUSE_USE_VERSION 26
#include <fuse.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

It is very clear that ***fuse.h*** is mandatory for implementing FUSE. Other three libraries required for using some functions such as ***fopen, fprint, strcat, strcpy, strcmp, memset, malloc*** etc. Using these libraries is are advantage of developing on user-space. Then, we used ***fuse_operation*** structure which contains definition of functions in FUSE. In ***fuse_operation*** structure, there is a lot of function can be used such as ***mkdir, chmod, open, read, release, rename*** etc. However, these functions are not mandatory to use and we only used functions that is needed with respect to the project. Actually, we wrote a struct for using these functions in way that mapping our functions we wrote to the these functions on the ***fuse_operation*** structure. These part of code is given below:

```
static struct fuse_operations tpc_operations = {
        .getattr    = tpc_getattr,
        .readdir    = tpc_readdir,
        .read        = tpc_read,
      }
```

Actually, we should use the ***unlink*** function to delete a file and ***rename*** function to change to name of the file, yet we could not finish on time project unfortunately. We could implement only ***getattr***, ***readdir*** and ***read*** functions.

Then, we wrote the main function to which runs our FUSE. It is given Figure 3 below:

```
int main(int argc, char *argv[]) {

    return fuse_main(argc, argv, &tpc_operations, NULL);
   }
```

In order to use data which is read from CSV file systematically, it was required to wrote a structure which contains variables such as city, district, neighborhood etc. We wrote ***tpc*** structure that ***tpc*** represents “Turkish postal codes”. This part of code is given below:

```
static const int linenumber=36307;//36307

struct tpc {
    char code[6];
    char neighborhood[30];
    char city[20];
    char district[30];
    char latitude[10];
    char longtitude[10];
    char text[200];
    char cityCode[3];
};

struct tpc *objs;

static const char *folder_path = "/NAMES";
static const char *folder_path2 = "/CODES";
```

Structure of ***tpc*** contains all columns in CSV file as variables. In addition we created ***text*** variable which is used for displaying data in a form of key-value pairs for each column when file is opened. Moreover, we created ***cityCode*** to store first two digits of the postal codes which represent the city. Their size are given according to the maximum number of characters of the column values. It is reasonable that type of longitude and latitude can be double, yet we prefer character array because they are not used in a numeric manner in this project.

There are other global variables with ***tpc*** structure above. In order to end some loop and read file, ***linenumber*** variable is used which represents the number of lines in CSV file. On the other hand, to create the CODES and NAMES directories in top of the file system, I used ***folder_path*** and ***folder_path2*** named variables.

Then we wrote the reading file part and assigned the each data on line in CSV file to the variables in the ***tcp*** structure. We wrote this part in main function and this part of code is given below:

```
int main(int argc, char *argv[]) {
    /*Memory allocation for the all objects,
     Actuall object number is 36307, yet we allocate a little bit more*/
    objs = malloc(36319 * sizeof(struct tpc)); //36307

    FILE *infile;   //for reading file, create input file
    //declarations of some requried varibles for some operations
    char line[100];
    char *templine,*linePtr;

    int i = 0;  /* i will correspond to each line and each object which are
mapping to each other*/

    //open file which will be read and read each line in a loop
    if ((infile = fopen("postal-codes.csv", "r")) != NULL) {

        while(fgets (line, 100, infile) != NULL){   /* 100 corresponds to
number of characters in a line*/

            linePtr = line;
            linePtr = strsep (&linePtr, "\n"); /* get line until come
across "/n and assign it to linePtr */

   /* Seperate line by "\t" and assign each value to related variable of
object.
       Do it for each line and each object. In this way construct the each
object of structure */
            templine=strsep(&linePtr,"\t");
            strcat(objs[i].code,templine);

            templine=strsep(&linePtr,"\t");
            strcat(objs[i].neighborhood,templine);

            templine=strsep(&linePtr,"\t");
            strcat(objs[i].city,templine);

            templine=strsep(&linePtr,"\t");
            strcat(objs[i].district,templine);

            templine=strsep(&linePtr,"\t");
            strcat(objs[i].latitude,templine);

            templine=strsep(&linePtr,"\t");
            strcat(objs[i].longtitude,templine);
/* to creating a text variable for each object via using snprinf function*/
            snprintf(objs[i].text, sizeof(objs[0].text),
                     "code: %s\nneigborhood: %s\ncity: %s\ndistrict:
%s\nlatitude: %s\nlongtitude: %s\n",
                     objs[i].code, objs[i].neighborhood, objs[i].city,
objs[i].district, objs[i].latitude,
                     objs[i].longtitude);
/*result of this function in such a form:
code: 34398
neighborhood: Maslak
city: Istanbul
district: Sariyer
```

```
latitude: 41.1083
longitude: 29.0183
*/
            /*get the first two digit of postal code and assign it to the
citycode*/
            strncpy(objs[i].cityCode, objs[i].code, 2);

             i++;

        }
    }
    else {          // if file is not found or not opened

        printf("File is not found!!");
    }

    return fuse_main(argc, argv, &tpc_operations, NULL);
}
```

In this part of code above, we read the CSV file and assign each data related variable of object. To obtain line, ***fgets*** function; to separate the line by new line and column (“\n”and ”\t”) ***strsep*** function; to assign value to object variable, ***strcat*** function; to create temp variable in a determined form, ***snprintf*** function; to copy first two digit of postal code to city code ***strncpy*** function is used. In a loop, file is read line by line and all lines separated one by one so that each data assign to variable of object. Variables are assigned to ***text*** and ***cityCode*** variables after some modifications on the strings. Detailed description of code is given with comment line above. I want to point out that we did reading file operation in a main function. However, it is required that any changes to the CSV file will be instantly visible through the FUSE-based file system and vice versa according to the project description. This is not possible with our method. We should write ***open*** function and reading file operation should be done in this function. In this way, at the each calling of open function reading operation will work and updates are done here instead of reading file once in main function as our wrote. Unfortunately, we noticed proper method after submission of the project.

Then we wrote ***tpc_getattr*** function which is one the most important function in FUSE. Main purpose of ***getattr*** function is to determine whether the given path which is one of the parameter of function, is a directory or file. In addition , to determine permissions about the file or directories, ***getattr*** function is used. Our ***tpc_getattr*** function and its detailed explanations with comment lines are given below:

```
static int tpc_getattr(const char *path, struct stat *st) {
//1st parameter is path: path of the file
//second paramerer is stat structure contain file's attributes
    /*clear the content of stat buffer */
    memset(st, 0, sizeof(struct stat));

    int i=0;

    /*Firstly we checked the root directory
       If the path is root  */
    if (strcmp(path, "/") == 0) {
        /* st_mode determine if the path is folder or regular file
         and it determine permissions of file
```

```
         S_IFDIR macro means path is directory
         S_IFREG macro means pat is regular file
         0755 is permission bits and
         it means only owner has authoritiy to write whereas
         the group ant others can oly read and execute directory
        */
        st->st_mode = S_IFDIR | 0755;
        /*st_link determine the number of hardlinks
          2 corresponds to directory itself and parent directory
        in other words "./" and "../"
            Totally links are these two and other subdirectories*/
        st->st_nlink = 2;
    }
    /*Secondly we checked if the path is "/NAMES" */
    else if (strcmp(path, folder_path) == 0) {
        st->st_mode = S_IFDIR | 0755;
        st->st_nlink = 2;
    }
     /*Thirdly, we checked if the path is "/CODES " */
    else if (strcmp(path, folder_path2) == 0) {
        st->st_mode = S_IFDIR | 0755;
        st->st_nlink = 2;
    }
    /* Until now path names are specified in advance, but now
     path names is nor known in advance. It is up to file has been read*/
    else{
        while(i<linenumber){ // linenumber=36307
          /* Firstly we create the all possible path
            can corresponds to directories and files
            in a loop check all objects */
            char *districtdir;
            districtdir=(char*)malloc(sizeof(char)*50); //memory allocation
            memset(districtdir,0,sizeof(char)*50);  //clear the content
just in case

            char *citydir;
            citydir=(char*)malloc(sizeof(char)*30);
            memset(citydir,0,sizeof(char)*30);

            char *neighborhooddir;
            neighborhooddir=(char*)malloc(sizeof(char)*60);
            memset(neighborhooddir,0,sizeof(char)*60);

            strcat(citydir,folder_path);
            strcat(citydir,"/");
            strcat(citydir,objs[i].city);
       /*    EXAMPLE:   "/NAMES/Istanbul"   */
            strcat(districtdir,citydir);
            strcat(districtdir,"/");
            strcat(districtdir,objs[i].district);
     /*    EXAMPLE  :"/NAMES/Istanbul/Sariyer"   */
            strcat(neighborhooddir,districtdir);
            strcat(neighborhooddir,"/");
            strcat(neighborhooddir,objs[i].neighborhood);
    /*    EXAMPLE:  "/NAMES/Istanbul/Sariyer/Maslak"   */
            char *cityCodedir;
            cityCodedir = (char *) malloc(sizeof(char) * 10);
            memset(cityCodedir, 0, sizeof(char) * 10);

            char *codedir;
            codedir = (char *) malloc(sizeof(char) * 18);
```

```
            memset(codedir, 0, sizeof(char) * 18);

            strcat(cityCodedir, folder_path2);
            strcat(cityCodedir, "/");
            strcat(cityCodedir, objs[i].cityCode);
        /*    EXAMPLE:  "/CODES/34"   */
            strcat(codedir,cityCodedir);
            strcat(codedir, "/");
            strcat(codedir, objs[i].code);
        /*    EXAMPLE:  "/CODES/34/34398"   */

            /* According to the project description,
             neighborhood and neighborhood postal codes should be file
                city and district and citycode should be directory*/

            if(strcmp(path,neighborhooddir)==0||strcmp(path,codedir)==0){
                /*S_IFREG means Regular file*/
                /*S_IFREG means Regular file*/
                 /*0444 means permission for only reading*/
                st->st_mode = S_IFREG | 0444;
                // number of link =1 beacuse it is file
                st->st_nlink = 1;
                // st_size is size of the file in terms of byte
                st->st_size = (off_t)strlen(objs[i].text);
                 break;
                /*  if path is found break the loop*/
            }

            else if(strcmp(path,citydir)==0||strcmp(path,districtdir)==0
               ||strcmp(path,cityCodedir)==0){

                st->st_mode = S_IFDIR | 0755;
                st->st_nlink = 2;
                 break;
            }

            i++;
        }
    }

    return 0;
}
```

In ***tpc_getattr*** function get the attributes of requested path, yet our method in ***tpc_getattr*** function is not good in terms of time complexity because when checking if the path is directory or file, it traverses all objects until found the path and did lots of modifications on strings for each loop. We submit our project as above yet, it can be ***tpc_getattr*** optimized such below:

```
static int tpc_getattr(const char *path, struct stat *st) {

    memset(st, 0, sizeof(struct stat));

    int i=0;
    int freq=0;

    char tempPath[60];
    memset(tempPath, 0,60);
    strcpy(tempPath,path);

    /*find the frequency of '/' in path */
```

```
    for(i = 0; tempPath[i] != '\0'; ++i)
    {
        if('/'== tempPath[i])
            ++freq;
    }
    char *parentPath = strtok(tempPath, "/");


    if (strcmp(path, "/") == 0) {
        st->st_mode = S_IFDIR | 0755;
        st->st_nlink = 2;
    }
    else if (strcmp(path, folder_path) == 0) {
        st->st_mode = S_IFDIR | 0755;
        st->st_nlink = 2;
    }
    else if (strcmp(path, folder_path2) == 0) {
        st->st_mode = S_IFDIR | 0755;
        st->st_nlink = 2;
    }
/* EXAMPLE: /NAMES/Istanbul/Sariyer/Maslak    */
    else if(strcmp(parentPath, folder_path+1) == 0 && freq==4){
        st->st_mode = S_IFREG | 0444;
        st->st_nlink = 1;
        st->st_size = (off_t)strlen(objs[i].text);

    }
/* EXAMPLE: /CODES/34/34398    */
    else if(strcmp(parentPath, folder_path2+1) == 0 && freq==3){
        st->st_mode = S_IFREG | 0444;
        st->st_nlink = 1;
        st->st_size = (off_t)strlen(objs[i].text);
    }
    else{
        st->st_mode = S_IFDIR | 0755;
        st->st_nlink = 2;
    }

    return 0;
   }
```

Instead of checking all object in loop, just count the number of '/' character in path, we can determine if the path is directory or file. Because we knew from the project description, which subdirectories should be file or directories. Only there are two cases for file, otherwise path will be directories and in this way we get rid of the loop. Therefore, performance of the program can get better.

Then, we wrote ***readdir*** function of FUSE. This function reads the directory and create the subdirectories or files for given path. In similar way to ***getattr*** function, we firstly read directories which is determined in advanced such as root (mounted directory), "NAMES" and "CODES". Then, we create the all possible path can corresponds to directories and files in a loop by checking all objects. Detailed code description and ***tpc_readdir*** function is given below:

```
static int tpc_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
                       off_t offset, struct fuse_file_info *fi) {

    int i = 0;
    // check if the path=root
    if (strcmp(path, "/") == 0) {
```

```
        filler(buf, ".", NULL, 0); //  Current Directory
        filler(buf, "..", NULL, 0); // Parent Directory
        filler(buf, folder_path + 1, NULL, 0);// create "/NAMES"
subdirectory"
        filler(buf, folder_path2 + 1, NULL, 0);// create "/CODES"
subdirectory"
    }

    // if path="/NAMES" create all cities
    else if (strcmp(path, folder_path) == 0) {
         i = 0;
        char tempCity[20]; // is used for checking city name is changed or
not
        memset(tempCity,0,sizeof(char)*20);

        filler(buf, objs[i].city, NULL, 0); // create first city in /NAMES
dir.
        strcpy(tempCity, objs[i].city);
    //traverse on all object if name of city is changed, create new city
        while (i < linenumber) {

            if (strcmp(tempCity, objs[i].city) == 0) {
            // if city is equals to temp city do nothing
                i++;

            } else {    //otherwise create th city and update the tempcity
                filler(buf, objs[i].city, NULL, 0);
                memset(tempCity,0,sizeof(char)*20);
                strcpy(tempCity, objs[i].city);
                i++;
            }

        }
    }
    //if path="/CODES" create all city codes
    // Strategy is the same as if path="/NAMES" create all cities above
    else if (strcmp(path, folder_path2) == 0) {
        pathIsFound=true;
        int i = 0;
        char tempCityCode[3];
        memset(tempCityCode,0,sizeof(char)*3);

        filler(buf, objs[i].cityCode, NULL, 0); //create first citycode in
/CODES dir.
        strcpy(tempCityCode, objs[i].cityCode);

        while (i < linenumber) { //36327

            if (strcmp(tempCityCode, objs[i].cityCode) == 0) {

                i++;

            } else {
                filler(buf, objs[i].cityCode, NULL, 0);
                memset(tempCityCode,0,sizeof(char)*3);
                strcpy(tempCityCode, objs[i].cityCode);
                i++;
            }

            // i++;
        }
```

```
    }

/* if path = city directory as an example: "/NAMES/Istanbul"
    create all districts of this city*/

    else {
        i=0;
         //traverse on all object if name of city is changed, create new
city
        while (i < linenumber) {
            /* create required varibles and
             form all possible directories for each iteration */
            char tempDistrict[30];
            memset(tempDistrict, 0, sizeof(char) * 30);
            strcat(tempDistrict, "");

            char *citydir;
            citydir = (char *) malloc(sizeof(char) * 30);
            memset(citydir, 0, sizeof(char) * 30);

            char tempCity[20];
            memset(tempCity, 0, sizeof(char) * 20);
            strcpy(tempCity, objs[i].city);

            strcat(citydir, folder_path);//folder_path =/NAMES
            strcat(citydir, "/");
            strcat(citydir, objs[i].city);
            /*EXAMPEL:/NAMES/Istanbul */
        //  if path = city directory as an example: "/NAMES/Istanbul
            if (strcmp(path, citydir) == 0) {
              // crate all districts of city until city is changed
                while (!(strcmp(tempCity, objs[i].city))) {

                    if (strcmp(tempDistrict, objs[i].district) == 0) {
                        i++;
                        // if distrcit are same do nothing
                    } else {// otherwise create district and update
tempDistrict
                        memset(tempDistrict, 0, sizeof(char) * 30);
                        strcpy(tempDistrict, objs[i].district);
                        filler(buf, objs[i].district, NULL, 0);
                        i++;
                    }
                }
                // if path is found and all districts are created break the
loop
                break;

            }

            i++;
        }
    }


    /* if path = district directory as an example:
"/NAMES/Istanbul/Sariyer"
        create all neighborhoods of this district*/
    /* Strategy is similar above if path = city directory as an example:
"/NAMES/Istanbul*/
        i=0;
```

```
         //traverse on all object if name of city is changed, create new
city
        while (i < linenumber) {
            /* create required varibles and
             form all possible directories for each iteration */
            char tempNeighborhood[40];
            memset(tempNeighborhood, 0, sizeof(char) * 40);

            char tempDistrict[40];
            memset(tempDistrict, 0, sizeof(char) * 40);
            strcat(tempDistrict, "");

            char *districtdir;
            districtdir = (char *) malloc(sizeof(char) * 50);
            memset(districtdir, 0, sizeof(char) * 50);

            strcpy(tempDistrict, objs[i].district);

            strcat(districtdir, folder_path);
            strcat(districtdir, "/");
            strcat(districtdir, objs[i].city);
            strcat(districtdir, "/");
            strcat(districtdir, objs[i].district);
            /*EXAMPEL:/NAMES/Istanbul/Sariyer */
        //  if path = city directory as an example:
"/NAMES/Istanbul/Sariyer
            if (strcmp(path, districtdir) == 0) {
                // crate all neighborhood of district until distrcict is
changed
                while (!(strcmp(tempDistrict, objs[i].district))) {

                    if (strcmp(tempNeighborhood, objs[i].neighborhood) ==
0) {
                        i++; // if neighborhood is the  same do nothing
                    } else {// otherwise create neigborhood and update
tempNeighborhood
                        memset(districtdir, 0, sizeof(char) * 40);
                        strcpy(tempNeighborhood, objs[i].neighborhood);
                        filler(buf, objs[i].neighborhood, NULL, 0);
                        i++;
                    }
                }
            // if path is found and all neighborhood are created break the
loop
                break;
            }
            i++;

        }
   // }
        i = 0;
/* if path = citycode create all postal codes for this city,
    as an example if path="CODES/34" create all postal codes for 34*/
   /* Strategy is the same as creating district for
    if path = city directory as an example: "/NAMES/Istanbul*/

    while (i < linenumber) {

        char tempCode[20] = "";
        char *cityCodedir;
```

```
        cityCodedir = (char *) malloc(sizeof(char) * 20);
        memset(cityCodedir, 0, sizeof(char) * 20);
        char tempCityCode[20];

        strcpy(tempCityCode, objs[i].cityCode);
        //

        strcat(cityCodedir, folder_path2);
        strcat(cityCodedir, "/");
        strcat(cityCodedir, objs[i].cityCode);

        if (strcmp(path, cityCodedir) == 0) {

            while (!(strcmp(tempCityCode, objs[i].cityCode))) {

                if (strcmp(tempCode, objs[i].code) == 0) {
                    i++;
                } else {
                    strcpy(tempCode, objs[i].code);
                    filler(buf, objs[i].code, NULL, 0);
                    i++;
                }
            }

           strcpy(tempCityCode, objs[i].cityCode);
            break;
        }

        i++;
        }
   // }
    return 0;
}
```

Finally we wrote the read function for reading files. According to project description, each district directory will show its neighborhood as text file and each city codes subdirectory will show the its neighborhood code as text file in a determined form. Text is stored in a ***text*** variable in ***tpc*** structure in a requested form. We checked the path for all possible directories by traversing on objects. When we found the path, related ***text*** variable of object is send to ***memcpy*** function as an parameter. Therefore, requested file is read. Read operation is similar to our previous project implementing a character devices. Our tpc_read function and its detailed explanations with comment lines are given below:

```
static int tpc_read(const char *path, char *buf, size_t size, off_t offset,
                    struct fuse_file_info *fi) {

    int i = 0;
    char *tempText;// is used when the path is found

    tempText = (char *) malloc(sizeof(char) * 200);
    memset(tempText, 0, sizeof(char) * 200);

    /* strategy is similar to readdir function
        */
    while(i<linenumber){
        /*traverse on all object and create required varibles and
             form all possible directories for each iteration */
    char *districtdir;
    districtdir = (char *) malloc(sizeof(char) * 50);
```

```
    memset(districtdir, 0, sizeof(char) * 50);

    char *citydir;
    citydir = (char *) malloc(sizeof(char) * 30);
    memset(citydir, 0, sizeof(char) * 30);

    char *neighborhooddir;
    neighborhooddir = (char *) malloc(sizeof(char) * 60);
    memset(neighborhooddir, 0, sizeof(char) * 60);

    strcat(citydir, folder_path);
    strcat(citydir, "/");
    strcat(citydir, objs[i].city);

    strcat(districtdir, citydir);
    strcat(districtdir, "/");
    strcat(districtdir, objs[i].district);

    strcat(neighborhooddir, districtdir);
    strcat(neighborhooddir, "/");
    strcat(neighborhooddir, objs[i].neighborhood);
    //EXAMPLE:  "/NAMES/Istanbul/Sariyer/Maslak"
    char *cityCodedir;
    cityCodedir = (char *) malloc(sizeof(char) * 10);
    memset(cityCodedir, 0, sizeof(char) * 10);

    char *codedir;
    codedir = (char *) malloc(sizeof(char) * 18);
    memset(codedir, 0, sizeof(char) * 18);

    strcat(cityCodedir, folder_path2);
    strcat(cityCodedir, "/");
    strcat(cityCodedir, objs[i].cityCode);

    strcat(codedir, cityCodedir);
    strcat(codedir, "/");
    strcat(codedir, objs[i].code);
//EXAMPLE:  "/CODES/34/34398"
        //There are only two cases for files
    if (strcmp(path, neighborhooddir) == 0 ) {

        strcat(tempText,objs[i].text);
        break;

    }
    else if(strcmp(path, codedir) == 0)
    {
        strcat(tempText,objs[i].text);
        break;
    }
    i++;
}
    //for reading this text we are copying these content to buffer by using
memcpy
    //start from the offset and read up to size
    memcpy(buf, tempText + offset, size);

    return strlen(tempText) - offset;   // return number of bytes have been
read
}
```

Consequently, we achieved the success to implement a FUSE file system which is constructed as an correct hierarchy and features according to project description. However, we could finish on time for doing delete and rename operations unfortunately is spite of the fact that we had started to implement project early. For us, it have been took so much time to wrote ***tpc_readdir*** function. After submission we did some optimizations on project, yet we demonstrate original project at the demo session. For test part, some screenshots are given Figure 1- Figure 5 below:

```
okan@okan-VirtualBox:~/Desktop$ fusermount -u /home/okan/Desktop/fusetest/
okan@okan-VirtualBox:~/Desktop$ gcc -g main.c -o main -D_FILE_OFFSET_BITS=64 -lfuse
okan@okan-VirtualBox:~/Desktop$ ./main /home/okan/Desktop/fusetest -s
okan@okan-VirtualBox:~/Desktop$ grep main /etc/mtab
main /home/okan/Desktop/fusetest fuse.main rw,nosuid,nodev,user=okan 0 0
okan@okan-VirtualBox:~/Desktop$ cd fusetest/
okan@okan-VirtualBox:~/Desktop/fusetest$ ls
CODES  NAMES
okan@okan-VirtualBox:~/Desktop/fusetest$ cd NAMES/
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES$ ls
Adana           Ankara     Bartin   Bolu       Denizli     Erzurum     Hatay          Karabuk    Kirikkale   Kutahya       Mus       Sakarya   Sivas     Van
Adiyaman        Antalya    Batman   Burdur     Diyarbakir  Eskisehir   Igdir          Karaman    Kirklareli  Malatya       Nevsehir  Samsun    Tekirdag  Yalova
Afyonkarahisar  Ardahan    Bayburt  Bursa      Duzce       Gaziantep   Isparta        Kars       Kirsehir    Manisa        Nigde     Sanliurfa Tokat     Yozgat
Agri            Artvin     Bilecik  Canakkale  Edirne      Giresun     Istanbul       Kastamonu  Kktc        Mardin        Ordu      Siirt     Trabzon   Zonguldak
Aksaray         Aydin      Bingol   Cankiri    Elazig      Gumushane   Izmir          Kayseri    Kocaeli     Mersin(Icel)  Osmaniye  Sinop     Tunceli
Amasya          Balikesir  Bitlis   Corum      Erzincan    Hakkari     Kahramanmaras  Kilis      Konya       Mugla         Rize      Sirnak    Usak
```

Figure 1 Compiling, running and displaying directories in root and NAMES

```
okan@okan-VirtualBox:~/Desktop/fusetest$ cd CODES/
okan@okan-VirtualBox:~/Desktop/fusetest/CODES$ ls
01  03  05  07  09  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75  77  79  81
02  04  06  08  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40  42  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80  99
okan@okan-VirtualBox:~/Desktop/fusetest/CODES$ cd 34
okan@okan-VirtualBox:~/Desktop/fusetest/CODES/34$ ls
34010  34075  34110  34147  34188  34218  34275  34325  34363  34387  34421  34460  34515  34550  34664  34699  34728  34755  34779  34825  34860  34893  34940  34990
34015  34076  34112  34149  34192  34220  34281  34330  34365  34394  34425  34464  34517  34555  34668  34700  34730  34758  34782  34829  34862  34896  34944
34020  34077  34116  34153  34194  34225  34283  34335  34367  34396  34427  34467  34520  34557  34672  34704  34732  34760  34785  34830  34865  34899  34947
34025  34080  34120  34158  34196  34230  34285  34337  34371  34398  34430  34470  34522  34560  34674  34707  34734  34762  34788  34840  34870  34903  34950
34035  34083  34122  34160  34197  34235  34287  34340  34373  34400  34433  34473  34524  34562  34676  34710  34736  34764  34791  34841  34873  34906  34953
34040  34087  34126  34164  34200  34240  34290  34342  34375  34403  34435  34480  34528  34570  34680  34714  34738  34766  34794  34843  34876  34909  34956
34045  34091  34130  34165  34203  34245  34295  34345  34377  34406  34437  34488  34530  34580  34682  34716  34740  34768  34799  34844  34880  34912  34959
34050  34093  34134  34173  34204  34250  34303  34347  34379  34408  34440  34490  34535  34582  34684  34718  34742  34771  34800  34846  34882  34916  34970
34055  34096  34140  34180  34209  34255  34307  34349  34380  34410  34445  34494  34537  34584  34688  34720  34744  34773  34805  34848  34885  34920  34973
34060  34098  34142  34182  34210  34260  34310  34353  34381  34413  34450  34500  34538  34590  34690  34722  34746  34774  34810  34852  34887  34925  34975
34065  34104  34145  34183  34212  34265  34315  34357  34382  34415  34453  34510  34540  34660  34692  34724  34750  34775  34815  34854  34888  34930  34977
34070  34107  34146  34186  34214  34270  34320  34360  34384  34418  34457  34513  34545  34662  34696  34726  34752  34776  34820  34858  34890  34935  34980
okan@okan-VirtualBox:~/Desktop/fusetest/CODES/34$
```

Figure 2 Displaying directories in CODES

```
okan@okan-VirtualBox:~/Desktop/fusetest$ cd NAMES/
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES$ cd Istanbul/
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul$ ls
Adalar      Bagcilar      Bayrampasa  Beyoglu       Esenler   Gaziosmanpasa  Kartal       Sancaktepe  Silivri      Tuzla
Arnavutkoy  Bahcelievler  Besiktas    Buyukcekmece  Esenyurt  Gungoren       Kucukcekmece Sariyer     Sisli        Umraniye
Atasehir    Bakirkoy      Beykoz      Catalca       Eyup      Kadikoy        Maltepe      Sariyer     Sultanbeyli  Uskudar
Avcilar     Basaksehir    Beylikduzu  Cekmekoy      Fatih     Kagithane      Pendik       Sile        Sultangazi   Zeytinburnu
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul$ cd Sariyer/
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul/Sariyer$ ls
Ayazaga  Maslak
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul/Sariyer$
```

Figure 3 Displaying directories in Istanbul and files in Sariyer

```
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul/Sariyer$ cat Maslak
code: 34398
neigborhood: Maslak
city: Istanbul
district: Sariyer
latitude: 41.1083
longtitude: 29.0183
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul/Sariyer$ od -j 26 -N 6 -a Maslak
0000032   a   s   l   a   k  nl
0000040
okan@okan-VirtualBox:~/Desktop/fusetest/NAMES/Istanbul/Sariyer$
```

Figure 4 Reading Maslak file and using some standart commands

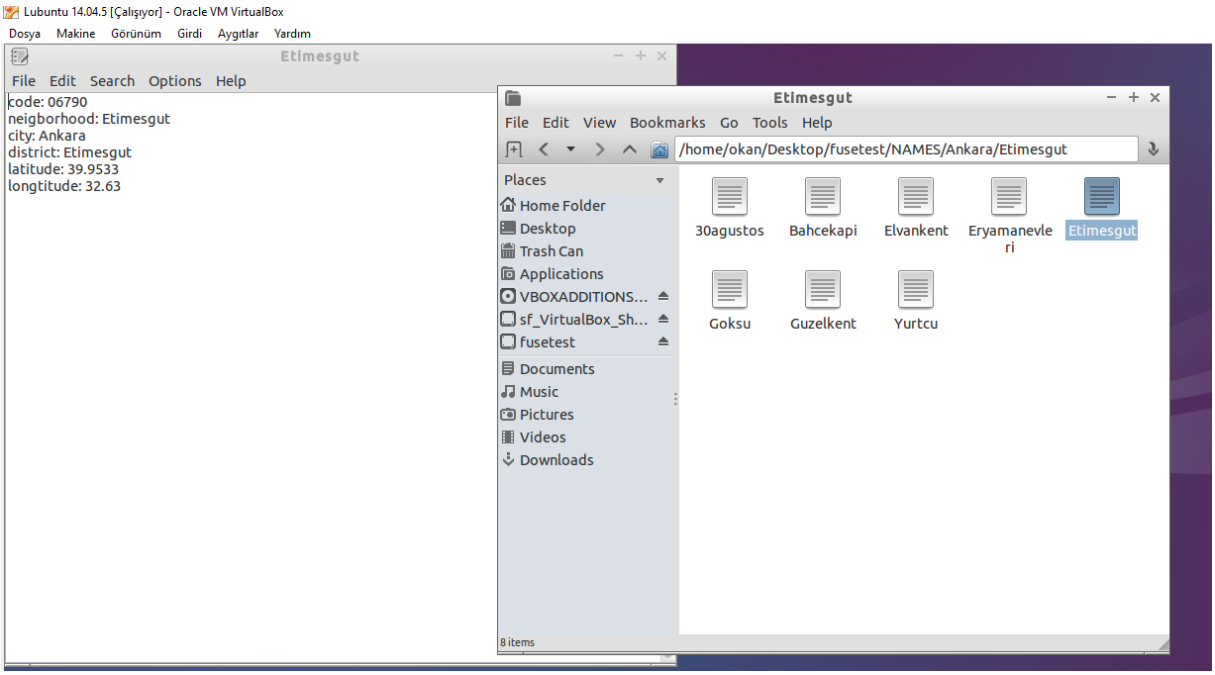

Figure 5 Example on file manager

3. Conclusion

It is inevitable fact that this project beneficial in terms of many aspect such as understanding the file system in OS , even implementing a file system by using FUSE and learning the distinction between working on user-space and kernel space. We faced many problems throughout the construction of the project, yet we overcome many of them and this way we developed our problem solving skills. I wish I was able to accomplish to complete the project with implementing rename and delete part.