LABORATORY 3

–

OPERATING SYSTEMS

RUFINO GARCIA SANCHEZ

Task 1. System calls and library functions - comparison of efficiency of system calls and library functions

Aim: ***Develop a program which compares efficiency of system calls and library functions for handling I/O operations.***

•Description: Write two programs:

- **Task1a which creates a file with the size given as and argument**

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

int main(void)
{
    int x;
    int i;
    FILE *fp;
    char ch [400];

    //We open the FILE
    fp = fopen("file.txt", "w+");
    printf("Introduce the size (it will be in Bytes) of the file you want and press ENTER: \n");
    scanf("%d",&i);
    fseek(fp, i , SEEK SET);
    fputc('\0', fp);

    //printf("Introduce the size of the file you want: \n");
    //scanf("%",i);


    if(fp == NULL) {
        printf("Error!");
        exit(1);
    }

    //Introduce some data in the FILE
    //printf("Enter data... (if you click on ENTER or SPACE you will finish writting): \n");
    //scanf("%s",ch);
    //fprintf(fp,"%s", ch);


    // closing the file pointer
    fclose(fp);

    return 0;
}
```

```
rufino@rufino:~/Documents/OS/Lab3$ ./task1a
Introduce the size (it will be in Bytes) of the file you want and press ENTER:
2048
```

This is my task1a.c. In this program we can observe how we put the size we want.

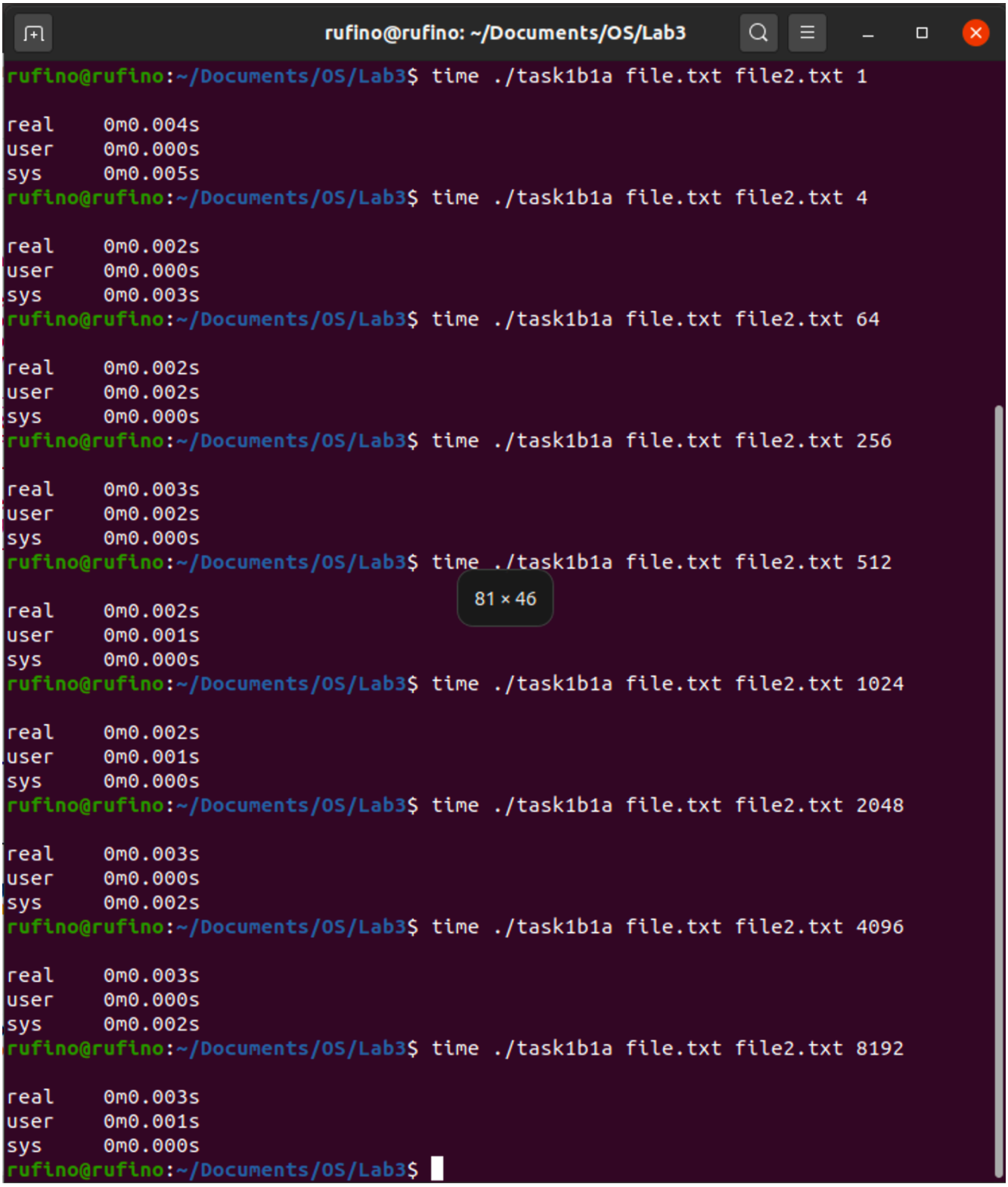
- **Task1b which copies this file a number of times specified by one of the arguments. This program should be written in two versions:**

  - **(i) using *read* and *write* system calls**
  - **(ii) using *fread* and *fwrite* library functions**

```c
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
  if (atoi(argv[4]) == 1) //We are using the function atoi in the library stdio.h to CONVERT the char* to int
  {
    int fd1, fd2;
    long int numbytes;
    char buffer[atoi(argv[3])];  //We are using the function atoi in the library stdio.h to CONVERT the char* to int

    if(((fd1 = open(argv[1], O RDONLY)) == -1) || ((fd2=open(argv[2],O CREAT|O WRONLY|O TRUNC, 0700)) == -1)){
        perror("problem with file, doesnt exist");
        exit(1);
    }


    while((numbytes=read(fd1, buffer, atoi(argv[3]))) > 0){
        if(write(fd2, buffer, numbytes) != numbytes){
            perror("problem copying the file");
            exit(3);
        }
    }

    close(fd1);
    close(fd2);
  }

  if (atoi(argv[4]) == 2)
  {
    FILE *fsource= fopen(argv[1],"r");
    FILE *fdest=fopen(argv[2],"w");
    char buffer[atoi(argv[3])];
    size_t num;
      fseek(fsource, 0, SEEK SET);

    while((num=fread(buffer, 1, sizeof(buffer), fsource))>0){
    fwrite(buffer, 1, sizeof(buffer), fdest);}

    fclose(fsource);
      fclose(fdest);

  }
}
```

As we can observe in this case it depends of what the user will put un the terminal to execute the program.

•For both modes measure real, user and system times for following size of buffers: 1,4,64,256,512,1024,4096,8192 bytes.



This is with the times with a size of 2048 bites and using write and read systems calls.

```
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 1

real    0m15.274s
user    0m0.002s
sys     0m0.000s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 4

real    0m7.048s
user    0m0.002s
sys     0m0.000s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 64

real    0m11.319s
user    0m0.000s
sys     0m0.003s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 256

real    0m8.052s
user    0m0.000s
sys     0m0.002s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 512

real    0m18.536s
user    0m0.002s
sys     0m0.000s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 1024

real    0m6.753s
user    0m0.002s
sys     0m0.000s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 2048

real    0m6.753s
user    0m0.002s
sys     0m0.000s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 4096

real    0m9.889s
user    0m0.000s
sys     0m0.002s
rufino@rufino:~/Documents/OS/Lab3$ time ./task1b2 file.txt file2.txt 8192

real    0m6.907s
user    0m0.000s
sys     0m0.002s
```

We can observe that when the file is bigger the real time is less.

All of this is with a file size of 2048 bites.

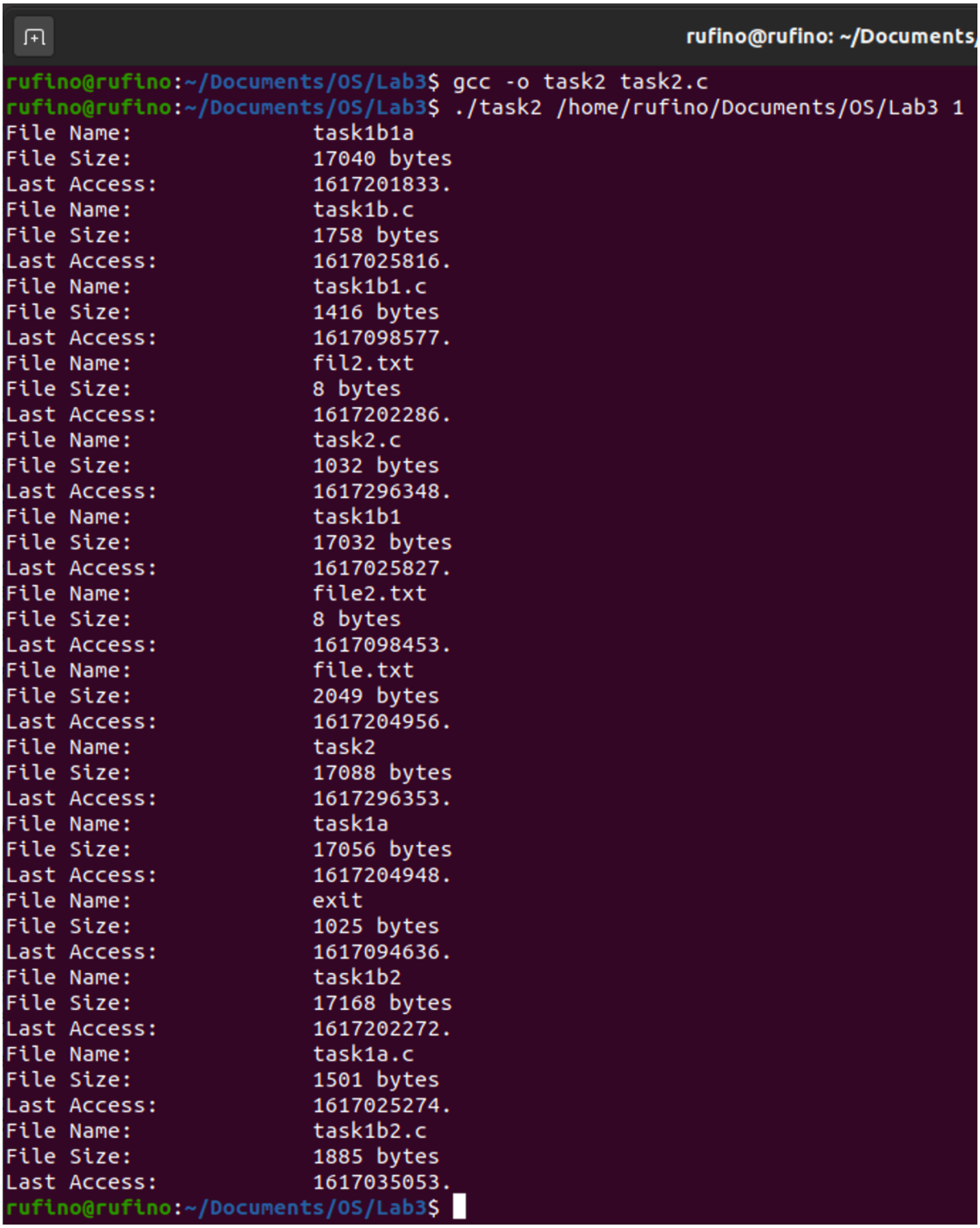
Task2. Browsing of directories

Write a program which browses a specified directory (given as the first argument of the program) and displays all names of files in this directory, their sizes and times of the last access and in nested directories. Only information about normal files should be displayed, links and special files should be omitted. Program should be written in two versions:

- **using opendir, readdir and stat function**

```
int task2a(DIR* dir){

    struct stat info;
    if (dir==NULL){
        return 1;
    }

    struct dirent* entity;
    entity=readdir(dir);
    while(entity !=NULL){
        if (!stat(entity->d name,&info))
        {
            if (S ISREG(info.st mode))
            {
              printf("File Name:              %s\n",entity->d name);
              printf("File Size:              %ld bytes\n",info.st size);
              printf("Last Access:            %ld. \n",info.st atime);
            }
            }
        entity=readdir(dir);
    }

    closedir(dir);
    return 0;
}

int main(int argc, char* argv[]){
    DIR *dirp;
    dirp = opendir(argv[1]);
    if(atoi(argv[2]) == 1){
        task2a(dirp);
    }
    else if(atoi(argv[2])==2){
        task2b(argv[1]);
    }
    return 0;
}
```

This is my fuction to search the directories using opendir, readdir and stat function.



As we can see works correctly.

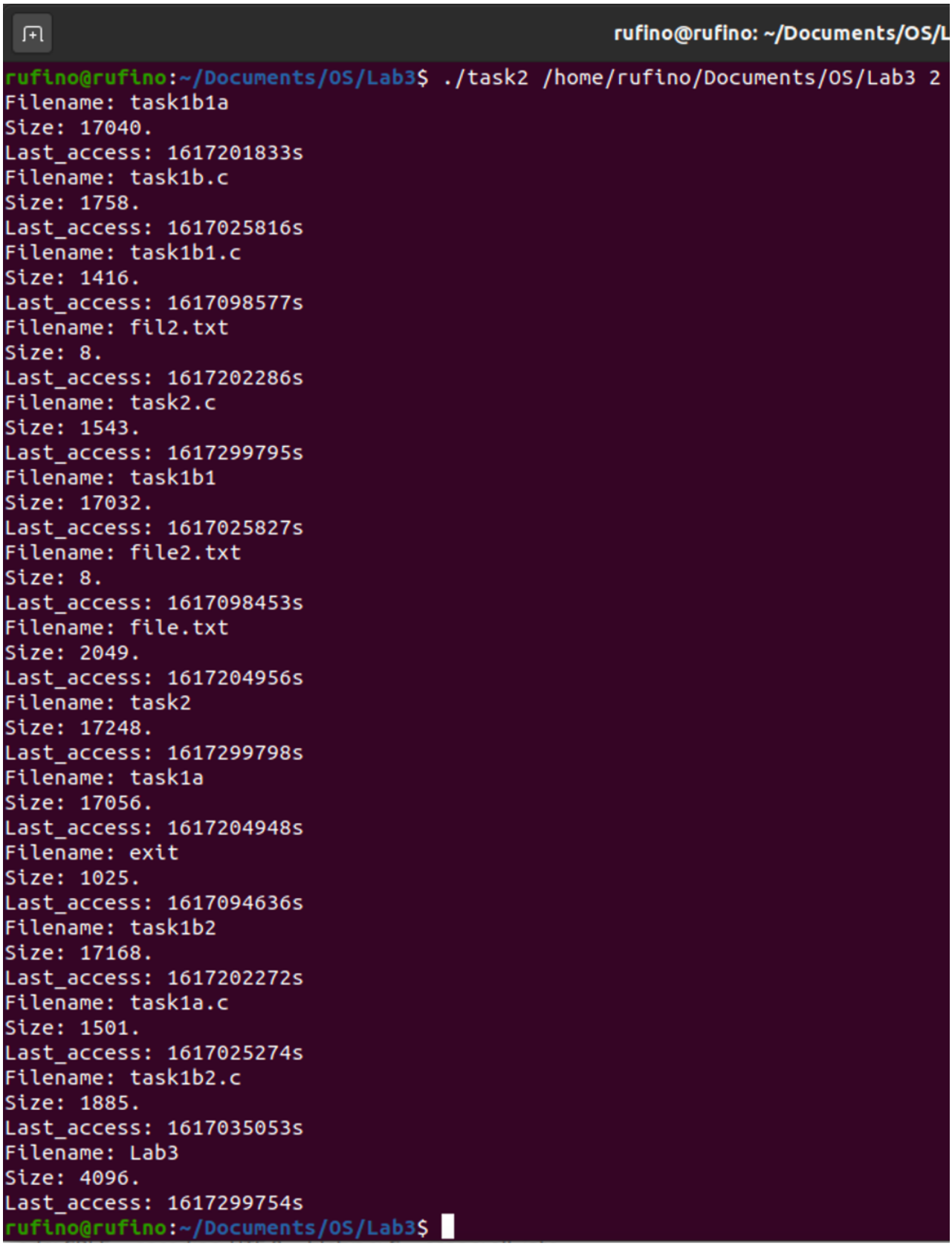
- **using nftw function**

This version should be chosen according to the second argument.

```c
#define  XOPEN SOURCE 500
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdint.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>
#include <ftw.h>

static int all info(const char *dir, const struct stat *sb, int flag, struct FTW *ftwbuff){
    if(flag == FTW DP || flag == FTW F){
        printf("Filename: %s\nSize: %ld.\nLast access: %llds\n", dir + ftwbuff->base,
        sb->st size,(long long)sb->st atime);
    }
        return 0;
}


int task2b(DIR *dir){
    int flag = 0;
    flag = FTW DEPTH;
    if(nftw(dir,all info,20,flag) == -1){

        perror("nftw");
        exit(EXIT FAILURE);
    }
    return 0;
}
```

As we can observe this is the code using nftw() function.

To work this part, I have to introcue **#define XOPEN_SOURCE_500**



Here we can obsere how the program is executed and how the files are displayed in the terminal like a tree.