CSE 344
FINAL REPORT

Ömer Faruk SAYAR
171044038

Server Side:

```c
pthread_mutex_t serverMutex;                    // Mutex for server directory updates
pthread_cond_t serverCond;                      // Condition variable to signal new client connection
pthread_t* threads;                             // Thread IDs of the server threads
int threadPoolSize = 0;                         // Number of server threads
int port = 0;                                   // Port number of the server
int logFD = 0;
int threadCount = 0;                            // Number of active server threads
char* directory = NULL;                         // Directory to be shared by the server
volatile sig_atomic_t shutdownServer = 0;       // Flag to indicate server shutdown
int clientSocketFD = 0;                         // Temporary client socket file descriptor
size_t max_entries = 8192;                      // Max number of entries in the directory (It is dynamically increased if needed)
```

These are the my global variables to provide program work correctly. I used one mutex for accessing the server directory and one condition variable to wake up a thread from thread pool when a client connected to server.

```c
void* workerThread(void* arg) {

    while(!shutdownServer){

        pthread_mutex_lock(&serverMutex);
        pthread_cond_wait(&serverCond, &serverMutex

        if (shutdownServer) {
            pthread_mutex_unlock(&serverMutex);
            threadCount--;
            pthread_exit(NULL);
        }

        int clientSocket = clientSocketFD;
        pthread_mutex_unlock(&serverMutex);

        int cs = 1;
        send(clientSocket, &cs,sizeof(int),0);
        ResponseType clientResponse;
```

```c
#ifndef COMMON_H
#define COMMON_H

#include <dirent.h>
#include <time.h>

#define MAX_PATH_LENGTH 8192

typedef enum ResponseType {
    OK,
    ERROR
}ResponseType;

typedef struct Info {
    unsigned char type;
    char path[8192];
    char name[8192];
    size_t size;
    int isDeleted;
    time_t lastModified;
} Info;

#endif /* COMMON_H */
```

On the worker threads, in a while loop first I read the server directory and determines the differences with one second frequency. If there any update on the server side, program creates an array contains of Info structs and writes to necessary information to these structs.

```c
/*-------------------------------------------------UPDATE CLIENT-------------------------------------------------*/

    //Previous State
    pthread_mutex_lock(&serverMutex);
    Info* dir = (Info*)malloc(entryCount * sizeof(Info));
    memset(dir, 0, entryCount * sizeof(Info));
    memcpy(dir, entries, entryCount * sizeof(Info));
    prevEntryCount = entryCount;
    pthread_mutex_unlock(&serverMutex);

    sleep(1);
    //Current State
    pthread_mutex_lock(&serverMutex);
    entryCount = 0;
    memset(entries, 0, max_entries * sizeof(Info));
    readDirectory(directory, entries, &entryCount);

    Info* diffArray = (Info*)malloc(max_entries * sizeof(Info));
    memset(diffArray, 0, max_entries * sizeof(Info));
    size_t diffCount = 0;

    //find only deleted files
    for (size_t i = 0; i < prevEntryCount; i++) {
        Info prevInfo = dir[i];
        int found = 0;
        for (size_t j = 0; j < entryCount; j++) {
            Info info = entries[j];
            if (strcmp(prevInfo.path, info.path) == 0) {
                found = 1;
                break;
            }
        }

        if (!found) {
            prevInfo.isDeleted = 1;
            diffArray[diffCount++] = prevInfo;
        }
    }

    //find new and modified files
    for (size_t i = 0; i < entryCount; i++) {
        Info info = entries[i];
        int found = 0;
        for (size_t j = 0; j < prevEntryCount; j++) {
            Info prevInfo = dir[j];
            if (strcmp(info.path, prevInfo.path) == 0) {
                found = 1;
                if (info.lastModified != prevInfo.lastModified && info.type == DT_REG) {
                    diffArray[diffCount++] = info;
                }
```

After how many Info will be sent is counted first I send the number to the client so that it can be receive the files/folders properly Then I send the file within 4096 bytes chunks.

```
printf("Sending %zu entries to client.\n", diffCount);

for (size_t i = 0; i < diffCount; i++) {
    Info info = diffArray[i];
    if (send(clientSocket, &info, sizeof(Info),0) == -1) {
        perror("Failed to send info to client");
        continue;
    }

    if (info.type == DT_DIR) {

        if (recv(clientSocket, &clientResponse, sizeof(ResponseType),0) == -1) {
            perror("Failed to read directory creation from client");
            continue;
        }
        if (clientResponse == OK) {

            if (info.isDeleted)
                printf("Directory '%s' deleted from the client.\n", info.name);

            else
                printf("Directory '%s' created on the client.\n", info.name);
        }
        else
    }

    else if (info.type == DT_REG) {
        // Send the file data
        recv(clientSocket, &clientResponse, sizeof(ResponseType),0);
        if(clientResponse == ERROR){
            continue;
        }

        if (info.isDeleted)
        {
            char log[256];
            strcpy(log, "File '");
            strcat(log, info.name);
            strcat(log, "' deleted from the client.");
            log_message(log);
            continue;
        }

        if (info.size == 0)
        {
            continue;
        }

        sendFile(clientSocket, &info);
```

```
void sendFile(int socketFD, const Info* info) {
    char path[MAX_PATH_LENGTH];
    strcpy(path, directory);
    strcat(path, info->path);
    FILE* file = fopen(info->path, "rb");
    if (file == NULL) {
        perror("Failed to open file");
        exit(1);
    }

    char log[MAX_PATH_LENGTH];
    strcpy(log, "File ");
    strcat(log, info->name);
    strcat(log, " last modified at ");
    strcat(log, ctime(&info->lastModified));
    log_message(log);

    char buffer[4096];
    size_t bytesRead;
    while ((bytesRead = fread(buffer, 1, sizeof(buffer), file)) > 0) {
        if (send(socketFD, buffer, bytesRead,0) == -1) {
            perror("Failed to send file data to client");
            exit(1);
        }
    }

    fclose(file);
}
```

After updates the clients according to changes on the server, server takes the update count from clients and does same operations as client. I also properly hande SIGINT signal and close the server and clients.

```
    /*-----------------------------------------------------UPDATE DIR-----------------------------------------------------*/
        if (send(clientSocket, &clientResponse, sizeof(ResponseType),0) == -1)
        {
            perror("Failed to send update response");
            break;
        }

        size_t entryNum = 0;
        ResponseType response;
        struct stat statbuf;

        if (recv(clientSocket, &entryNum, sizeof(size_t),0) <= 0) {
            cs = 0;
            break;
        }

        printf("Number of entries coming: %ld\n", entryNum);

        response = OK;
        pthread_mutex_lock(&serverMutex);
        if (send(clientSocket, &response, sizeof(ResponseType), 0) == -1) {
            perror("Failed to send response to client");
            cs = 0;
            break;
        }

        for (size_t i = 0; i < entryNum; i++){    …

        pthread_mutex_unlock(&serverMutex);
    }
```

Client Side:

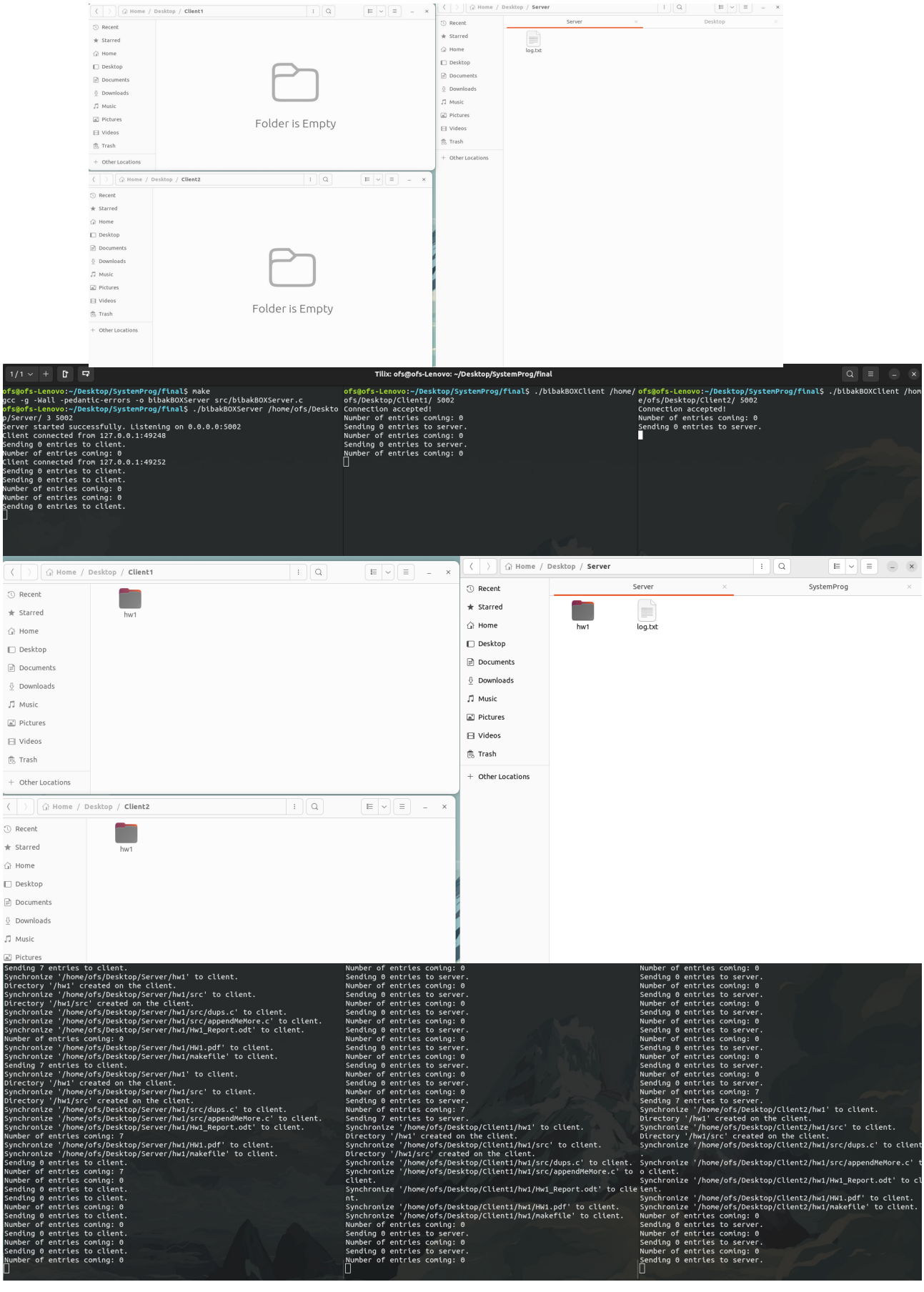
On the client side program does the same operations in the reverse order. First gets the updates from server then sends its updates to server.

```
while (!shutDown)
{
    struct stat statbuf;
    ResponseType response;
    size_t prevEntryCount;
    size_t entryNum = 0;

    if (recv(socketFD, &entryNum, sizeof(size_t),0) == -1) {…

    printf("Number of entries coming: %ld\n", entryNum);
    fflush(stdout);

    response = OK;
    if (send(socketFD, &response, sizeof(ResponseType), 0) == -1) {…

    for (size_t i = 0; i < entryNum; i++)
    {
        Info info;
        if(recv(socketFD, &info, sizeof(Info),0) == -1){
            response = ERROR;
            if(send(socketFD, &response, sizeof(ResponseType), 0) == -1){
                perror("Failed to send response to server");
                break;
            }
            response = OK;
            continue;
        }

        char path[MAX_PATH_LENGTH];
        strcpy(path, directory);
        strcat(path, info.name);

        if (info.type == DT_DIR)
        {
            if (info.isDeleted){
                char cmd[MAX_PATH_LENGTH];
                strcpy(cmd, "rm -rf ");
                strcat(cmd, path);
                system(cmd);
            }

            else{
                mkdir(path,0777);
                struct utimbuf new_times;
                stat(path, &statbuf);
                new_times.actime = statbuf.st_atime;  // Keep the current last access time
                new_times.modtime = info.lastModified;    // Set the new modified time
                utime(path, &new_times);
            }
```

```
else if (info.type == DT_REG)
{
    if (info.isDeleted == 1)
    {
        remove(path);
        if(send(socketFD, &response, sizeof(ResponseType), 0) == -1){
            perror("Failed to send response to server");
            break;
        }
        continue;
    }

    if (stat(path, &statbuf) == 0)
    {
        if (statbuf.st_mtime == info.lastModified)
        {
            response = ERROR;
            if(send(socketFD, &response, sizeof(ResponseType), 0) == -1){
                perror("Failed to send response to server");
                break;
            }
            continue;
        }
    }

    //read socket until the info.size reached and write to file
    int fd = open(path, O_WRONLY | O_CREAT | O_TRUNC, 0777);
    if(send(socketFD, &response, sizeof(ResponseType), 0) == -1){
        perror("Failed to send response to server");
        break;
    }

    if (info.size > 4096){
        int chunk = info.size / 4096;
        int remainder = info.size % 4096;
        for (int i = 0; i < chunk; i++)
        {
            char* buffer = (char*)malloc(4096);
            recv(socketFD, buffer, 4096, 0);
            write(fd, buffer, 4096);
            free(buffer);
        }
        char* buffer = (char*)malloc(remainder);
        recv(socketFD, buffer, remainder, 0);
        write(fd, buffer, remainder);
        free(buffer);
    }

    else if(info.size < 4096 && info.size > 0){
        char* buffer = (char*)malloc(info.size);
        recv(socketFD, buffer, info.size, 0);
        write(fd, buffer, info.size);
        free(buffer);
    }

    else{
        close(fd);
        continue;
    }
```

RESULTS

1- Update Clients with the server



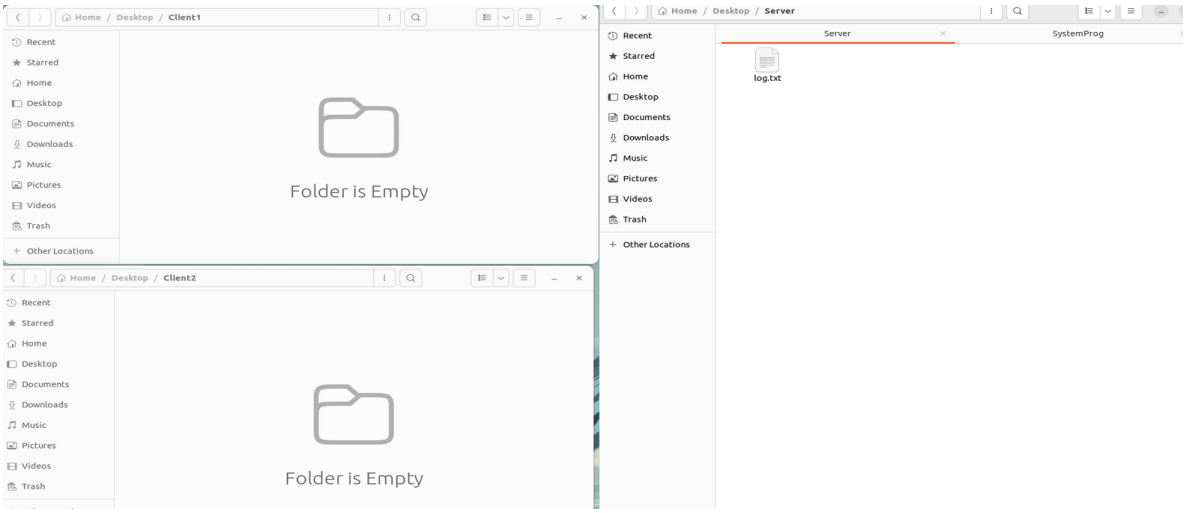
Open | log.txt ~/Desktop/Server | Save

```
[2023-06-16 23:17:27] File /hw1/src/dups.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/src/appendMeMore.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/Hw1_Report.odt last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/HW1.pdf last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/makefile last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/src/dups.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/src/appendMeMore.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/Hw1_Report.odt last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/HW1.pdf last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/makefile last modified at Fri Apr 14 23:16:48 2023
```

```
Synchronize '/home/ofs/Desktop/Server/hw1' to client.
Directory '/hw1' deleted from the client.
Synchronize '/home/ofs/Desktop/Server/hw1/src' to client.
Directory '/hw1/src' deleted from the client.
Synchronize '/home/ofs/Desktop/Server/hw1/src/dups.c' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/src/appendMeMore.c' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/Hw1_Report.odt' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/HW1.pdf' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/makefile' to client.
Sending 7 entries to client.
Synchronize '/home/ofs/Desktop/Server/hw1' to client.
Directory '/hw1' deleted from the client.
Synchronize '/home/ofs/Desktop/Server/hw1/src' to client.
Directory '/hw1/src' deleted from the client.
Synchronize '/home/ofs/Desktop/Server/hw1/src/dups.c' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/src/appendMeMore.c' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/Hw1_Report.odt' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/HW1.pdf' to client.
Synchronize '/home/ofs/Desktop/Server/hw1/makefile' to client.
Number of entries coming: 0
Sending 0 entries to client.
Number of entries coming: 7
Number of entries coming: 0
Sending 0 entries to client.
```

```
Sending 0 entries to server.
Number of entries coming: 0
Sending 0 entries to server.
Number of entries coming: 0
Sending 0 entries to server.
Number of entries coming: 7
Sending 7 entries to server.
Synchronize '/home/ofs/Desktop/Client1/hw1' to client.
Directory '/hw1' deleted from the client.
Synchronize '/home/ofs/Desktop/Client1/hw1/src' to client.
Directory '/hw1/src' deleted from the client.
Synchronize '/home/ofs/Desktop/Client1/hw1/src/dups.c' to client.
File '/hw1/src/dups.c' deleted from the client.
Synchronize '/home/ofs/Desktop/Client1/hw1/src/appendMeMore.c' to client.
File '/hw1/src/appendMeMore.c' deleted from the client.
Synchronize '/home/ofs/Desktop/Client1/hw1/Hw1_Report.odt' to client.
File '/hw1/Hw1_Report.odt' deleted from the client.
Synchronize '/home/ofs/Desktop/Client1/hw1/HW1.pdf' to client.
File '/hw1/HW1.pdf' deleted from the client.
Synchronize '/home/ofs/Desktop/Client1/hw1/makefile' to client.
File '/hw1/makefile' deleted from the client.
Number of entries coming: 0
```

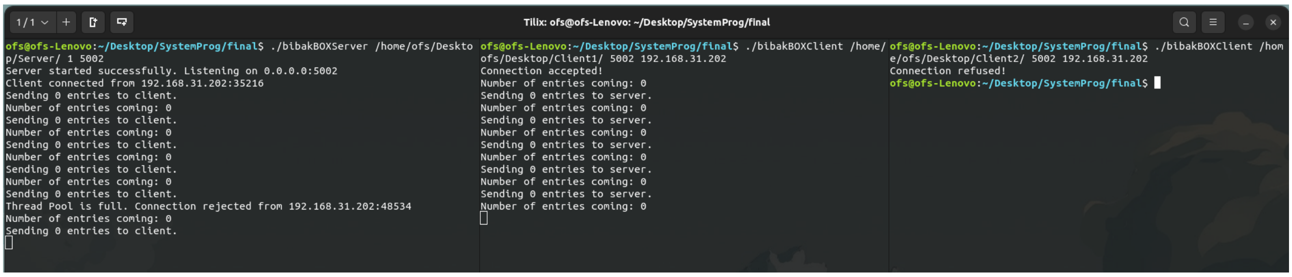
```
Sending 0 entries to server.
Number of entries coming: 0
Sending 7 entries to server.
Synchronize '/home/ofs/Desktop/Client2/hw1' to client.
Directory '/hw1' deleted from the client.
Synchronize '/home/ofs/Desktop/Client2/hw1/src' to client.
Directory '/hw1/src' deleted from the client.
Synchronize '/home/ofs/Desktop/Client2/hw1/src/dups.c' to client.
File '/hw1/src/dups.c' deleted from the client.
Synchronize '/home/ofs/Desktop/Client2/hw1/src/appendMeMore.c' to client.
File '/hw1/src/appendMeMore.c' deleted from the client.
Synchronize '/home/ofs/Desktop/Client2/hw1/Hw1_Report.odt' to client.
File '/hw1/Hw1_Report.odt' deleted from the client.
Synchronize '/home/ofs/Desktop/Client2/hw1/HW1.pdf' to client.
File '/hw1/HW1.pdf' deleted from the client.
Synchronize '/home/ofs/Desktop/Client2/hw1/makefile' to client.
File '/hw1/makefile' deleted from the client.
Number of entries coming: 7
Sending 0 entries to server.
Number of entries coming: 0
Sending 0 entries to server.
```

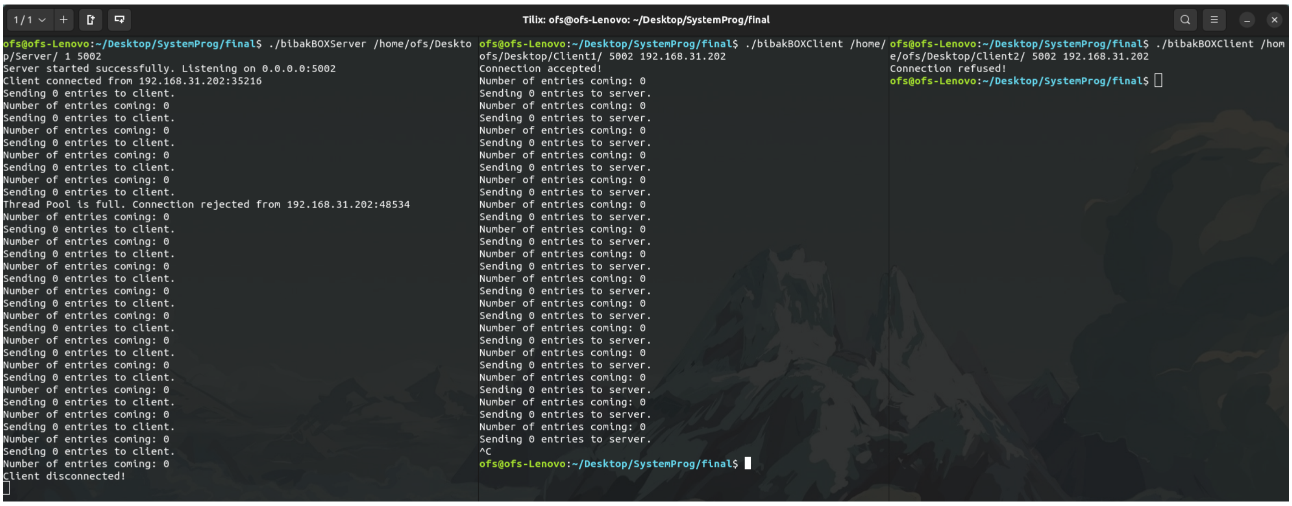


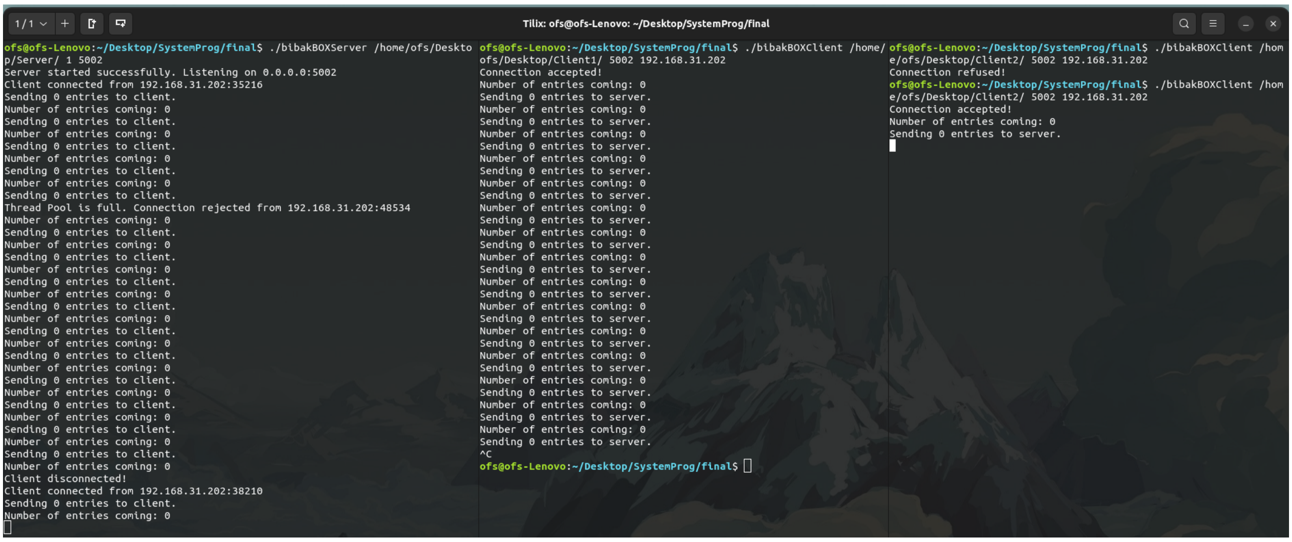
Open | log.txt ~/Desktop/Server | Save

```
[2023-06-16 23:17:27] File /hw1/src/dups.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/src/appendMeMore.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/Hw1_Report.odt last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/HW1.pdf last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:27] File /hw1/makefile last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/src/dups.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/src/appendMeMore.c last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/Hw1_Report.odt last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/HW1.pdf last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:17:28] File /hw1/makefile last modified at Fri Apr 14 23:16:48 2023

[2023-06-16 23:20:37] File '/hw1/src/dups.c' deleted from the server.
[2023-06-16 23:20:37] File '/hw1/src/appendMeMore.c' deleted from the server.
[2023-06-16 23:20:37] File '/hw1/Hw1_Report.odt' deleted from the server.
[2023-06-16 23:20:37] File '/hw1/HW1.pdf' deleted from the server.
[2023-06-16 23:20:37] File '/hw1/makefile' deleted from the server.
[2023-06-16 23:20:38] File '/hw1/src/dups.c' deleted from the client.
[2023-06-16 23:20:38] File '/hw1/src/appendMeMore.c' deleted from the client.
[2023-06-16 23:20:38] File '/hw1/Hw1_Report.odt' deleted from the client.
[2023-06-16 23:20:38] File '/hw1/HW1.pdf' deleted from the client.
[2023-06-16 23:20:38] File '/hw1/makefile' deleted from the client.
[2023-06-16 23:20:39] File '/hw1/src/dups.c' deleted from the client.
[2023-06-16 23:20:39] File '/hw1/src/appendMeMore.c' deleted from the client.
[2023-06-16 23:20:39] File '/hw1/Hw1_Report.odt' deleted from the client.
[2023-06-16 23:20:39] File '/hw1/HW1.pdf' deleted from the client.
[2023-06-16 23:20:39] File '/hw1/makefile' deleted from the client.
[2023-06-16 23:20:40] File '/hw1/src/dups.c' deleted from the server.
[2023-06-16 23:20:40] File '/hw1/src/appendMeMore.c' deleted from the server.
[2023-06-16 23:20:40] File '/hw1/Hw1_Report.odt' deleted from the server.
[2023-06-16 23:20:40] File '/hw1/HW1.pdf' deleted from the server.
[2023-06-16 23:20:40] File '/hw1/makefile' deleted from the server.
```

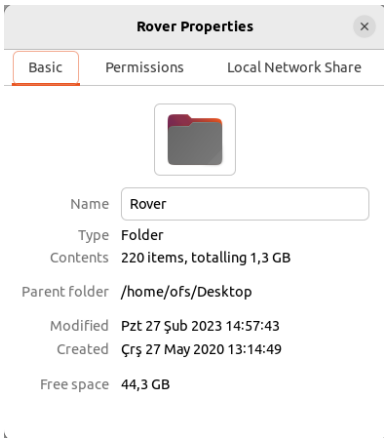
2- Disconnect/ Connect

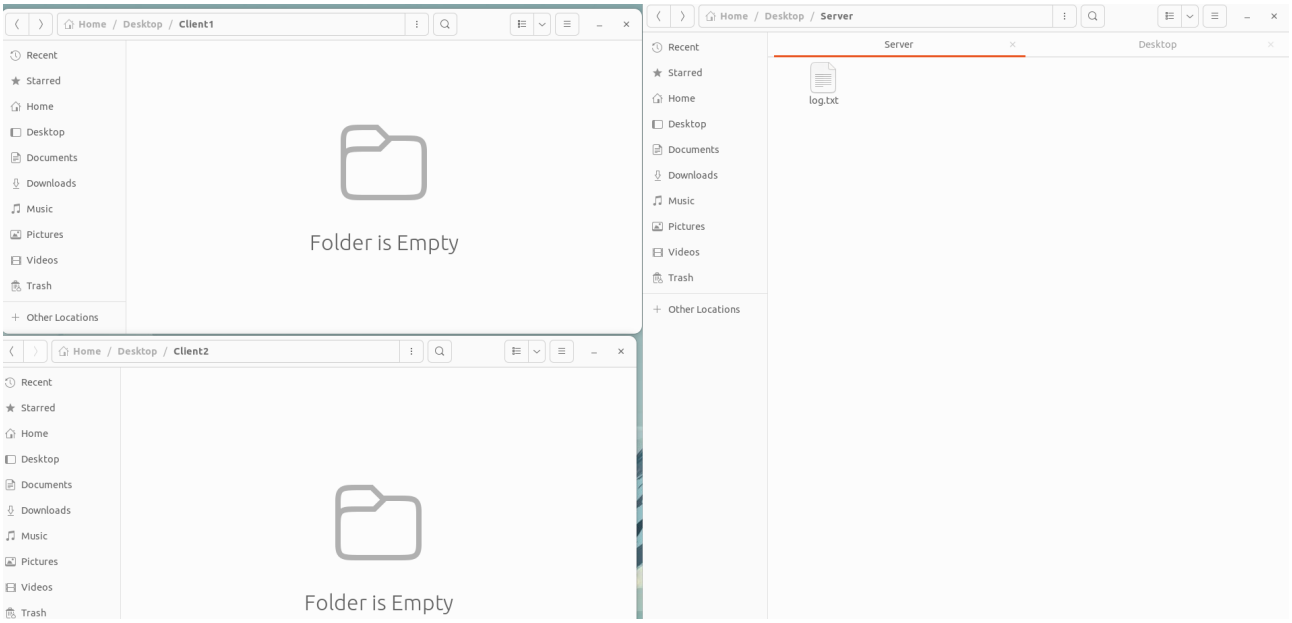


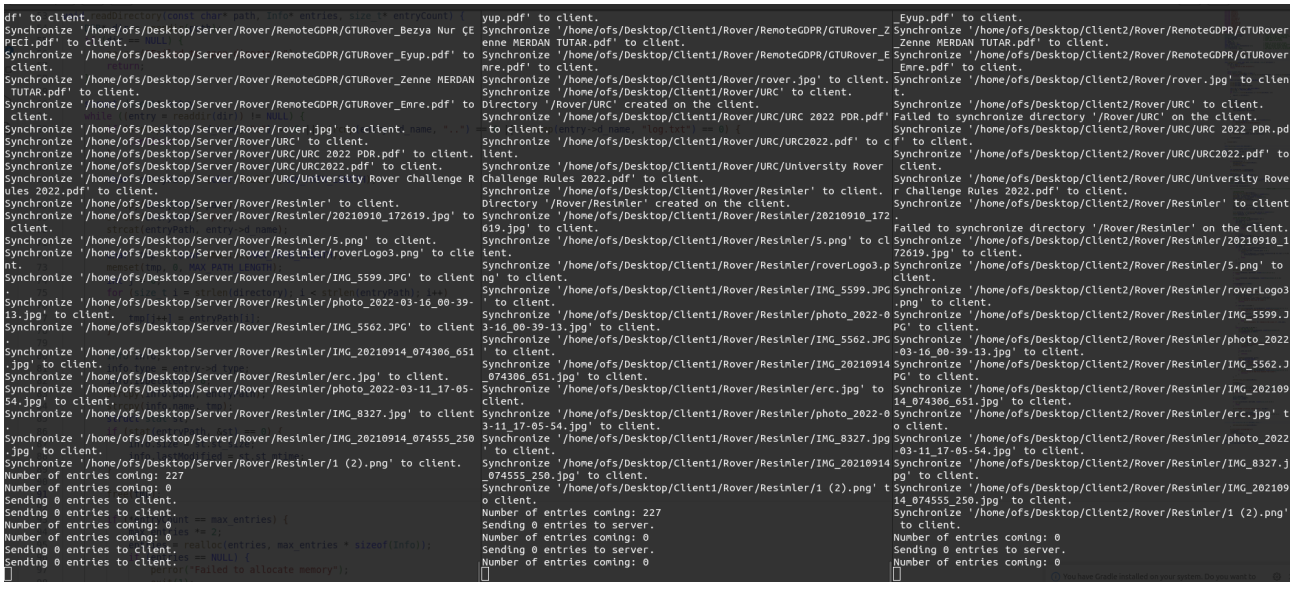


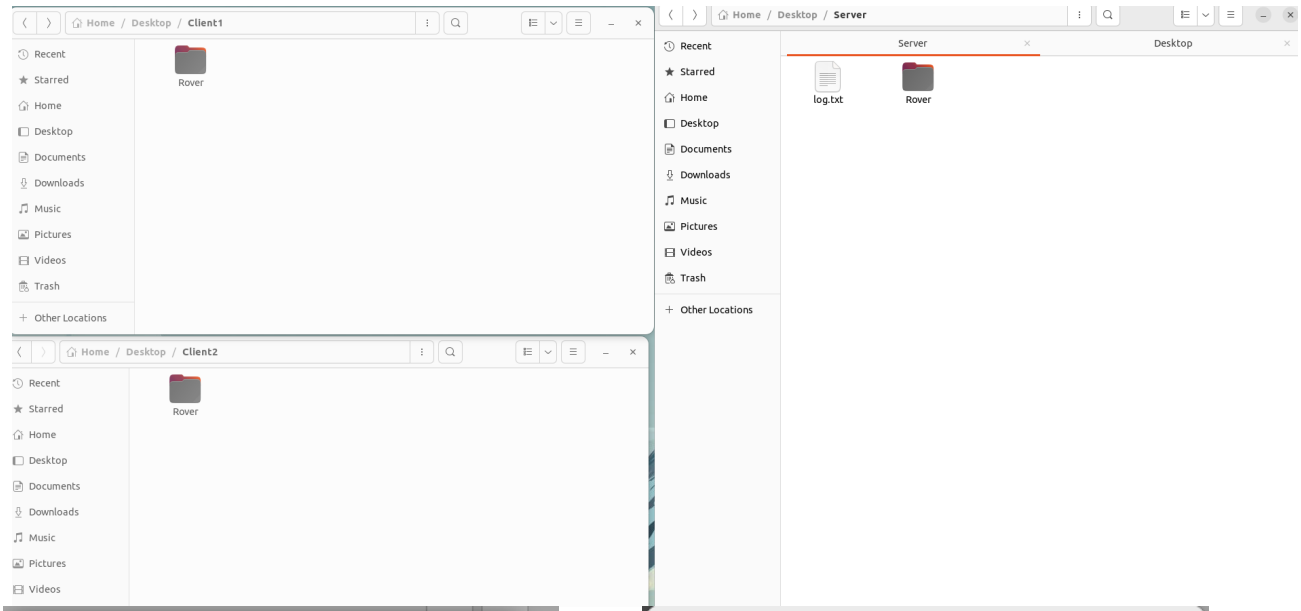


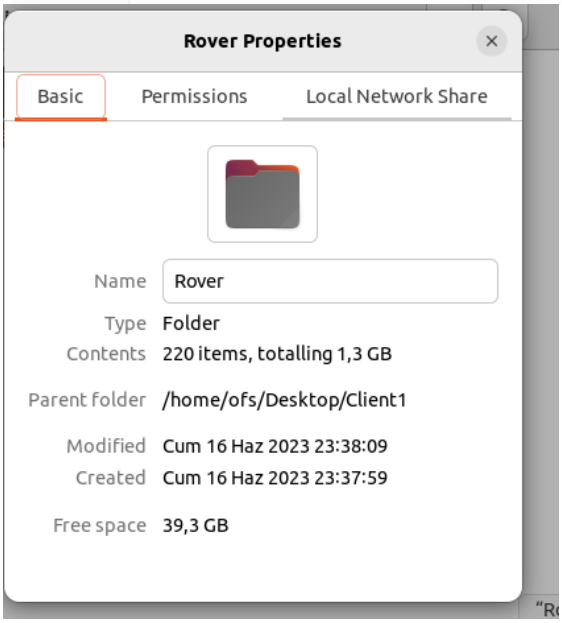
3- Sending 1GB over folder

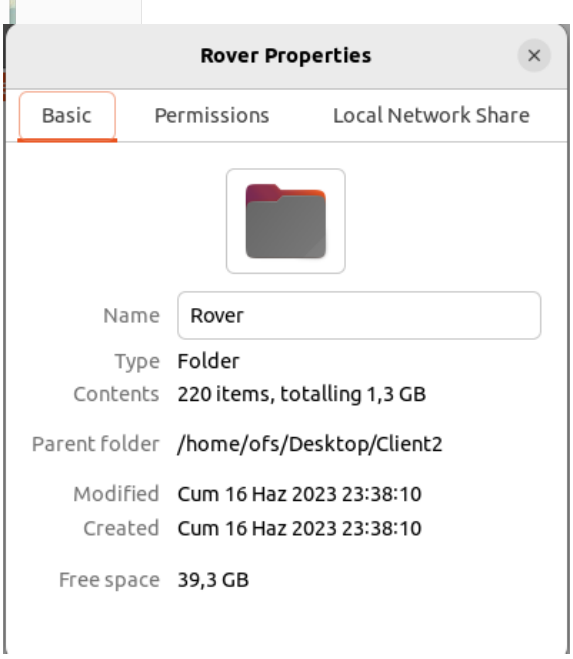








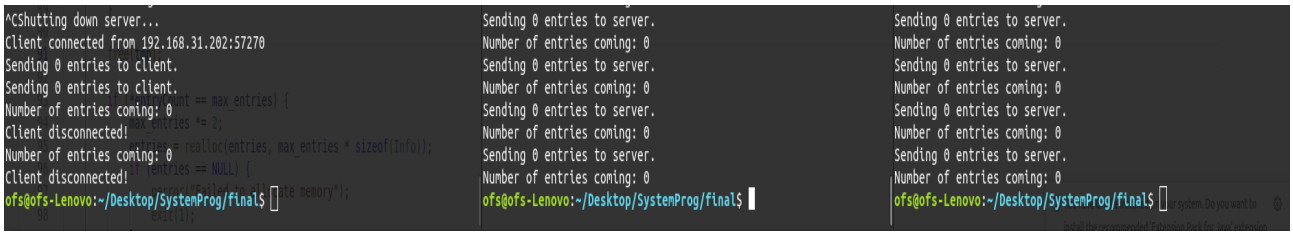




Open log.txt ~/Desktop/Server Save

```
[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/.mxproject last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/.settings/stm32cubeide.project.prefs last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/.settings/language.settings.xml last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/STM32F401RETX_RAM.ld last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Device/ST/STM32F4xx/License.md last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Device/ST/STM32F4xx/Include/stm32f401xe.h last modified at Sun Aug  7 14:35:32
2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Device/ST/STM32F4xx/Include/system_stm32f4xx.h last modified at Sun Aug  7
14:35:32 2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Device/ST/STM32F4xx/Include/stm32f4xx.h last modified at Sun Aug  7 14:35:32
2022

[2023-06-16 23:38:01] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/cmsis_compiler.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_sc300.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/cmsis_version.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/cmsis_armclang.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_sc000.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_cm0.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/cmsis_armcc.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_armv8mbl.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/cmsis_gcc.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_cm1.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_cm33.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/mpu_armv8.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_cm3.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/tz_context.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_cm4.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/core_cm7.h last modified at Sun Aug  7 14:35:32 2022

[2023-06-16 23:38:02] File /Rover/Drive-System-Stm32F411-main(1)/Drive-System-Stm32F411-main/Drivers/CMSIS/Include/cmsis_iccarm.h last modified at Sun Aug  7 14:35:32 2022
```

4 – Signal Handling



5- Valgrind