

Systems Software Report CA2

TU857

BSc in Computer Science

**Omair Duadu**

**C18322011**

School of Computer Science

TU Dublin – City Campus

**08/12/2021**

**Table of Contents**

*Functionality Checklist* 3

*Feature 1 - Client Program* 4

*Feature 2 – Server Program* 7

*Feature 3 - Multithreaded connections* 11

*Feature 4 - File Transfer* 12

*Feature 5 - Transfer Authentication using Real and Effective ID’s* 14

*Feature 6 - Synchronisation (Mutex Locks)* 17

*Conclusion* 17

# *Functionality Checklist*

|  |  |  |
| --- | --- | --- |
| ***Feature*** | ***Description*** | ***Implemented*** |
| F1 | Client | Yes |
| F2 | Server | Yes |
| F3 | Multithreaded connections | Yes |
| F4 | File Transfer | Yes |
| F5 | Transfer Authentication using Real and Effective ID’s | Yes |
| F6 | Synchronisation (Mutex Locks) | Yes |

Demo Video: https://www.youtube.com/watch?v=rwp9awKrris

# *Feature 1 - Client Program*

Detailed description of the implementation and architecture choices made for the client program.

*Step 1: Connecting to Server*

The client program has many different things function at the same time, this includes primitives, sock, connect, and sending & receiving.

The socket descriptor is created and then tries to establish a connection with a specific port on the server which was opened for this. Therefore, the server program needs to be running first as it allows the client to find what it is trying to connect to.

Text

Description automatically generated

*Image 1.1 Socket functionality*

*Step 2: User Credentials*

The next thing in order is to collect the userid, and groupid from the current user in the client terminal and store them in buffers, which will be sent to the server and printed on the server terminal for identification purposes.

Text

Description automatically generated

*Image 1.2 Credentials functionality*

*Step 3: File & Destination Selection*

In this section the user is told to enter in the name of the file they which to transfer and then given three different options on where they would like to transfer the file or whether to exit.

This is done using a classic switch statement with number inputs and can detect if an unavailable option was picked. This switch statement determines the path for the transfer and it will be stored in a buffer. The switch statement also includes an exit option to quit the program

Text

Description automatically generated

*Image 1.3 File & Destination Selection functionality*

*Step 4: Sending, checking, and validation*

After all the previous steps the file is finally sent to the server if it exists, an error will be printed if the file does not exist. If everything is successful, the file will be sent and a message will be displayed.

Text

Description automatically generated

*Image 1.4 File & Destination Selection functionality*

# *Feature 2 – Server Program*

Detailed description of the implementation and architecture choice for the server program. The lifeline of the communication between the client and the server program and how this is managed should be described in detail.

*Step 1: Accepting/Establishing Connection & Identification*

The server binds to a connection request from the port using a bind(), then the server will continu to listen to any incoming traffic and using a listen(), and finally it accepts the connections using the accept().

After establishing the connection the server will receive info about the UserID and the GroupID of the terminal of the connection client.

It will then print out the UserID and GroupID and username.

Text

Description automatically generated

*Image 2.1 Printing client user Data*

*Step 2: Changing Server User and Group ID*

The server will then do its own check by finding the group the using getGrouplist(). This is very important, as it will now adopt the userID and groupID for the server program, so that it can correctly transfer the files to these restricted directories with the new identity it has adopted. The server will then print the new id it has adopted.

Text

Description automatically generated

*Image 2.2 Finding Groups*

Text

Description automatically generated

*Image 2.3 changing server to client identity*

*Step 3: Doing the transfer*

With the new identity of the server, it is now free to complete the main objective of this program. It will receive the data and place it in the designated folder. There is error checking and displays in this process as well.

Text

Description automatically generated

*Image 2.4 transfer functionality*

# *Feature 3 - Multithreaded connections*

Describe how the socked server program has offered concurrent connections.

The system being multithreaded is essential in this use case and scenario it is very important that multiple user’s connection and transfers can occur otherwise he inefficiencies. Like in the previous CA the server will create a new thread of execution to handle the requests of each client every time the sever accepts a connection request from a client.

Text

Description automatically generated

*Image 3.1 Multithreaded functionality*

Each thread that the server creates performs its function and then terminates upon completion of its task, which is the when the client disconnects from the server.

# *Feature 4 - File Transfer*

Detailed description of the implementation process for sending a file from the client to the server.

*Step 1: Entering the file name*

On the client terminal the user enters the name of the file they want to transfer into a array

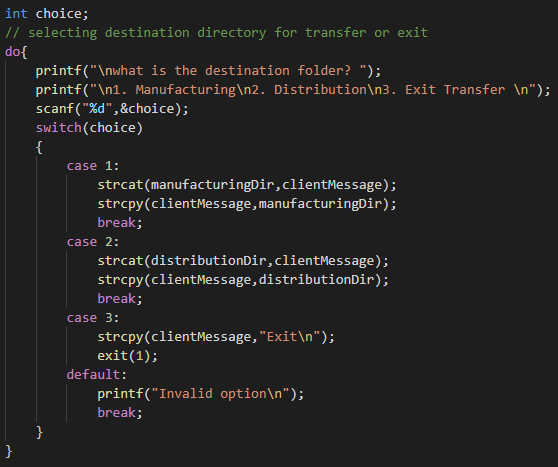
Text

Description automatically generated

*Image 4.1 File Selection*

*Step 2: Selecting the destination*

The user then chooses the destination from switch statement or can just exit and disconnect from the server. In each case there is a strcat() and strcpy(), which will store the names of the folder in a clientMessage array, and this will form a proper path to send the file through.



*Image 4.2 Directory Selection*

*Step 3: Retrieval and Transfer*

The client now retrieves the file from the folder for it to be transferred to the server.

Text

Description automatically generated

*Image 4.3 File Retrieval & Transfer*

*Step 4: Error checking and transfer*

Now the client will check if it can find the file that was chosen and send

Text

Description automatically generated

*Image 4.4 checking and transfer*

*Step 5: Error checking and receiving*

The server now receives the data and does multiple checks on the data received.

The checking is done through if statements to check size, by comparing block sizes, and also to check if any data was received at all. If no data is received errors will be displayed.

Text

Description automatically generated

*Image 4.5 checking and receiving*

# *Feature 5 - Transfer Authentication using Real and Effective ID’s*

Detailed description the process used to determine if a specific user is permitted to transfer a file to the Manufacturing/Distributions folders in the root Intranet folder.

*Step 1: UserID and GroupID collecting*

I have done the transfer of authentication using real and effective ID. The first thing that is done after connecting with the server is transferring over the UserID and GroupId.

This is accomplished by using getting the info by getuid() for the UserID and getgid() for the GroupID. It is then sent over to the server

Text

Description automatically generated

*Image 5.1 Getting and send User and Group ID’s*

*Step 2: Receiving ID*

After receiving the UserID and GroupID of the client the server it first displays the ID’s it has received and then changes its own ID’s to match the client’s.

Text

Description automatically generated

*Image 5.2 displaying the Id’s*

*Step 3: Server changing ID’s*

The server will now change the ID’s to that of the client, and this is of extremely high importance. The server program runs with root permissions and therefore can make any changes it wants, which is very bad security management as it might allow the distribution user to access and transfer to the manufacturing directory.

The users have already been placed into groups which have privileged access to their own department directories. Therefore, with this we can guarantee that users without the correct permissions cant access these directories.

Text

Description automatically generated

*Image 5.3 Changing and displaying Servers new Id’s*

*Step 4: Reverting ID*

After completing the transfer, the ID’s will be reverted, so that they can be changed again for the next user who connects through the client

*Image 5.4 Reverting ID’s*

Text

Description automatically generated

# *Feature 6 - Synchronisation (Mutex Locks)*

How synchronisation was achieved for the concurrent access to shared resources.

The Mutex lock is very important for locking the resources so that no user can access them while being used by the client, therefore if anyone else wanted to use the same resource they would have to wait for the mutex lock to be released first.

The mutex lock has been placed after the inputs so that the mutex lock disruption to resources would be at a very low minimum, so that multiple people can start the transfer until the final selection, therefore the impact is as low as possible.

It is then released after the server reverts the ID change and

Text

Description automatically generated

*Image 6.1 Reverting ID’s*

# *Conclusion*

Summary of the implementation and achievement

First the server opens a socket to bind and listen on. The client will then send a request to that socket where the server then accepts the connection. A connection is now established.

The server now creates a thread to preform the transfer, and the User ID and GroupID of the client is sent to server, where it will now adopt them.

The Client Now ask for a file name in the shared folder and then a destination directory of either Distribution or Manufacturing. File checking occurs ow to check if the file exists and if it doesn’t it will disconnect. The server now receives the folder and file.

The server now attempts to write to the folder however if the user is not a part of the permissions group it will fail with the server’s new ID’s.

If the client did receive a valid folder for the user, the server will successfully transfer the data to that folder and respond with a success or failure in the server terminal.

And final for the achievement I am very proud of being able to bring all the different functionality together to make a single working system which all meshes together successfully with the intended purpose behind it being achieved.

Additionally a particular point of the project which I am very happy about is the mutex lock which was designed to run in the perfect most efficient time I could identify, it could have been placed in other places both it wouldn’t have been as successful anywhere I else from my current experience.