

Systems Software Report CA2

TU856

BSc in Computer Science

**Philip Herweling**

**C18470774**

School of Computer Science

TU Dublin – City Campus

**10/12/2021**

**Table of Contents**

*Functionality Checklist* 3

*Feature 1 - Client Program* 4

*Feature 2 – Server Program* 4

*Feature 3 - Multithreaded connections* 4

*Feature 4 - File Transfer* 5

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

*Feature 6 - Synchronisation (Mutex Locks)* 5

*Conclusion* 5

# *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 |

Video Link(Also submitted the video as well):

<https://www.youtube.com/watch?v=fO7mLFUNZgU>

# *Feature 1 - Client Program*

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

For this section I am going to use bullet points in order to make it easier to follow my train of thought.

1. The first thing I done was get the user id, group id, e user id and e group id.
2. I then got the users, username by using getpwuid. The reason I want to get the username is because I will be passing it to the server along with the user id because I need both of these to find the groups they are linked to.
3. Next, I created the socket
4. After creating the socket I then used it to connect to the server
5. After connecting to the server I then proceeded to send over the user id and the username.
6. I then send over the file name to the server which I want to send to either the distribution or manufacturing directory. I then proceed to send over the file path to server which allows the server to send the file from the client folder to the distribution or manufacturing folder depending on the user group.

Note: Every time I sent the server information, I also received information back which let the client know weather to continue or if an error occurred.

# *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.

For this section I am going to use bullet points in order to make it easier to follow my train of thought.

1. The first thing I done was create my thread function this is where my socket , prepare my socket address and listen for any incoming connections from my client.
2. After that I then created a thread which accepts the connections from the client.
3. After I make a connection with the client I had to make a connection handler which handles the connection.
4. The first thing in my connection handler is where I take in the users username from the client. I then send back a received message to the client.
5. The second thing I do in my connection handler is take in the users username. I then send back a received message to the client.
6. After getting the username and user id I then find the groups that the user is associated with. I store the groups the user is associated with in an array.
7. I then have a for loop which displays the groups the user is associated with and also inside that for loop I have an if, else statement which sets the path where the file should be sent to, either to the distribution folder or the manufacturing folder.
8. Then I read in the file name and the path where the file is stored.
9. I then use mutex to lock the thread and I proceed to move the file to the location I set above.
10. I then change who owns the file to which ever user got it sent over.
11. After that I then unlock thread using mutex unlock.

Note: The same as the client every time the server file received something is sent a message back to the client.

# *Feature 3 - Multithreaded connections*

Describe how the socked server program has offered concurrent connections.

The socked server program that I created is a multithreaded program which contains two parts that run concurrently. Each thread has a separate path of execution. I achieved this by changing the path where the file is sent depending on the groups the user is associated with.

This allows my program to basically do multiple tasks at once.

*Feature 4 - File Transfer*

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

1. The first thing I done was send the file name and path from the client to the server
2. I took in the filename and file path in the server program
3. I then set the file path to the folder I want to send the file to.
4. I then wrote code to open the file, if it couldn’t be opened I printed out a message saying that the file couldn’t be opened but if it could be opened I then proceed to move the file from the client directory to either the manufacturing or distribution directory. I also changed the person who owns the file to who ever moved the file.

# *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 Marketing/Offers/Sales/Promotions folders in the root Intranet folder.

# *Feature 6 - Synchronisation (Mutex Locks)*

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

I used mutex in my program to lock down the shared resources when a transfer was happening. This was necessary because if a transfer was happening and another user tried to the same transfer at the same time it could affect the transfer process as the destination path could have changed.

# *Conclusion*

Summary of the implementation and achievement

In conclusion I thought this was a very enjoyable project, I learned a lot about client server programming and learned the hard way about how important the flow between them is. I thought the assignment was challenging but also reasonable. I think that after completing this assignment I have a lot better understanding of client server programming, synchronisation, multithreading and mutexes.