Systems Software Report CA2

DT211
BSc in Computer Science

Aaron Sharkey
C15350401

School of Computer Science
TU Dublin – City Campus

25/04/2019

Table of Contents

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

Feature 1 - Client Program

The client program allows users to transfer files to the server.

When the client is started it gathers the users name from the environment variables using getenv("user") and greets them. Once the name is gathered it used to gather a list of groups that the user is associated with on the system, these include the ones specified in the brief such as Sales, marketing and promotions. The user's id is also gathered.

After user information is gathered a connection is made to the server through a socket. After the socket connection is established a list of possible file transfer destinations is provided to the user based on the groups that they are apart of. For example, if the user is only apart of the sales group, the root directory and the sales folder will be options in the destination menu. The chosen destination is where the file will be placed on the server. This was done as the server and client reside on the same server so checking the user's permissions prior to file transfer enforces added security measures.

After the folder destination is chosen the user is prompted to provide the path to the file they wish to transfer. The user is then also prompted to give the file a new name, this will be the name of the file on the server. This new file name is combined with the destination they chose to provide a file path that will be used by server to store the file in the appropriate destination, it looks something like "Sales/salesinfo.txt". This was done to prevent confusion amongst files and allows users to better differentiate files.

After the user has entered the information previously their details are transferred to the server. The users ID, username and the file path are sent to the server and are stored. This information is used by the server to change ID and to place the transferred file in the appropriate directory.

Once the user's information has been sent to the server the file transfer begins. The file given by the user is transferred in blocks to the server over the socket. Once the transfer has concluded the client receives feedback from the user as to whether the file transfer was successful or not.

The socket is then closed, and the client exits.

Feature 2 – Server Program

The Server provides concurrent socket connections using threads.

When the server is started a socket is created and bound to port 9000. Once the server is bound it will listen for client connections.

When a client contacts the server, a connection is established and a thread is generated. The connected client's socket id is provided to the thread via pointer. Once the thread is started it gathers the clients socket id and begins listening to the connection.

The server-side code is executed inside of a while loop. This while loop checks to see if a client has disconnected or not by checking for a KILL message. Whenever a message is received from the client its size is verified, if the client has disconnected the KILL message is communicated and the client's connection will be closed by the server.

Once a client connection is fully established the server begins listening for information. The server first receives the file path for the file to be transferred, this is the concatenation of the new file path and the users chosen destination from the client and It looks like "Sales/salesinfo.txt". The server then receives the user's id and the users name so that it can change ids further on.

After the user information has been collected the server then begins receiving the file transfer. A temporary file is created in the tmp folder and is named tmp plus the user's ID for example "tmp1001". The file is received in chunks through the socket and written to the temporary file until the transfer has completed.

After the file has been saved the server changes the ownership of the temporary file from root to the current user. The Chown function is used for this.

Once the file owner has been changed the mutex is locked and the server changes real and effective ids to reflect the connected user.

The groups the user belongs to are gathered in a similar fashion to that which is implemented on the client and they are used to set the groups for the server. The servers effective group / user IDs are set using the information passed by the client.

Once the servers ID reflects the connected user a move command is issued. Move was used as it moves the file to another destination but can also renames it in the process. A move command is generated by concatenating the temporary file destination and the file path provided by the client. This move command is then executed and will move and rename the temporary file to the destination provided by the client if the user is apart of the appropriate groups. System was used as I encountered issues with the system ids being set using execlp despite it being more secure.

Feature 3 - Multithreaded connections

Concurrent connections are offered by the server through threads. Each connection to the server is managed by its own thread and offers multiple connections at a time. Once a client connects their ID is passed to a thread where the server-side logic is then carried out for each connection concurrently.

Feature 4 - File Transfer

The file transfer is done using the socket connection between the client and the server. The transfer begins when the user provides the path to file to be transferred. This file is read in chunks of 512 bytes, each of these chunks are then sent to the server through the socket. The server receives these chunks and writes them to a temporary file. Chunks are read and sent until the file has been completely read and transferred, then transfer ceases.

Feature 5 - Transfer Authentication using Real and Effective ID's

Authentication is performed on both client and server side.

The client-side checks users and the groups they belong to before providing file destinations. As the client resides on the same system and utilises the same users as the server this will prevent wasting resources on the server. When a user starts the client all their associated groups IDs are gathered in a list. This list is then checked for the group IDs that match the ID's specified in the brief. All matching ids are presented in a list of destinations from which the user can select. This is where their file will be stored on the server.

The server authenticates file transmissions by switching effective ids. When a user connects their username and user ID are sent to the server. After the users file has been transferred to the server and stored temporarily the server will modify its effective ids to the that of the users and attempt to move the temporary file to the destination specified by the client. All folders in the website are locked to their respective groups and only members or root can manipulate files within them. By changing ids and supplying the users groups the server can mimic the permissions of the user and ensure that the actions of the client are allowed given the permissions in place.

Feature 6 - Synchronisation (Mutex Locks)

Thread synchronisation is enforced through a mutex lock, Threads must secure the mutex when they need to switch ids and perform file manipulation. This stops issues arising when commands are being issued and when files are being moved around the system and ensures singular access to a resource.

Conclusion

The software solution outlined above provides a design and implementation that satisfies the brief. It ensures concurrency through threading, socket communication and proper Linux security authentication.

This project proved to be quite difficult however it was an incredibly useful learning experience that has sparked an interest in c programming again for me.