**Server Project**

This personal project has the objective of practicing the C language – the fields of file management and communication between server and client using the transmission control protocol (TCP) - the proper structure of a project – structuring the project in a way that is easy to maintain and compile using the Makefiles - testing said project – building testing files for the project.

This function of the server is to handle the various users connected to the server and handle the files of each user. A good example of how this project should resemble is much like a dropbox.

**Client Program**

The client side will consist of authenticating an account, contact the server and then receive a key. This key will bind the client program to the server so no other process can have two accounts in session at the same time.

In session, depending on the type of account, premium or not, the user can perform various actions such as saving or downloading files, check the amount of available space, the files it has stored, delete files or share them with another user.

After each command, there will be a local cache, to improve performance, thus reducing the number of requests to the server. This means there will be a log file, registering every command the user has made, and, after a while, updates the files on the server.

**Server Program**

The server side, as opposed of the client side, will be a conglomerate of independent programs. This means that the “server” has and independent program that handles requests, a program that handles with the users, and a program that handles the files, all working independently of each other. For example, if the program that manages files is down, the request handler and user handler will continue to function. Thanks to this independency from each service, it also means that is possible to change, for example, change the user service for another without changing the requests handler and the files handler.

**Client Program (In Depth)**

* **Authentication**

At the start of the program, the client requires to authenticate into the program using his credentials, such as an ID and a password. If both User ID and password are correct the output will be like:

User ID:

User ID: client

User ID: client  
Password:

User ID: client  
Password: 123

User ID: client  
Password: 123  
Authenticating…

User ID: client  
Password: 123  
Authenticating…  
Authenticated.  
Type “help” or “?” to see the available commands.  
client>

In case of the User ID or Password are incorrect the output will be:

User ID: client  
Password: 123  
Authenticating…  
Authentication failed. User ID or password are incorrect.  
User ID:

To exit the program, it is only needed to press enter when the program is still waiting for a user ID, such as:

User ID:

User ID: (Enter)

User ID:  
Goodbye!

* **In Session**

When the client is in session, the client can user various commands, such as;

/- help – Displays the commands available  
/- ? – Displays the commands available  
/- acst – Displays the information of the account currently registered  
/- files – Displays the names of the files on the user disk  
/- add – Adds a new file to the disk  
/- grab – downloads a file from the disk  
/- rm – Removes a file  
/- mv – moves a file to a new location  
/- cp – copies the file to a new location  
/- mkdir – creates a new directory  
/- rmdir – removes a directory  
/- sadd – Shares a file or directory with a different user  
/- srm – Removes a shared User from a file or directory  
/- sfiles – Displays the files or directories currently being shared