

Network Attached Storage in C with socket (server/client)

Prof.  Gilberto Echeverria Furo

Enrique Lira Martinez A01023351

Emiliano Abascal Gurría A01023234

Cesar Armando Valladares Martinez A01023503

**Description:**

The program created is a network-attached storage making a file-level computer data storage server connected to a computer network providing data access to a certain group of clients. The NAS is specialized for serving files by its software so we are going to make a program in which you as a client can interact with the server and send file or data to be store in the server.

They typically provide access to files using network file sharing protocols. NAS devices began as a convenient method of sharing files among multiple computers.The benefits of dedicated network-attached storage, compared to general-purpose servers also serving files, include faster data access, easier administration, and simple configuration.

The program would require a log in for security reasons, when a client logs in, then it would have access to it’s assigned directory, in it, the client will be able to create, delete or modify directories, and upload or download files into it’s home directory or subdirectories created by the user.

Also the program have sharing functions so the user can decide if they want to share a certain directory with other users, with the permissions that the owner decides to give to the invited client (read, write, download or upload).

This would function over command line.

**Functions:**

char \* Encrypt\_password(char s[])

This function recives an original string (passwoord) and is responsible for encrypting ti at the time of registration, it is also used at the login, all stored passwords are encrypted.

int checkValidAccount(int account,char password[])

This is the function that is used at the login, it is responsible for checking that the user ID exists, and if it existsm compares of the encrypted passwords, if both conditions are true, the session is started without problems.

int SaveUser(thread\_data\_t\* data ,char password[])

When registering a new user, a unique ID is created that is delivered to the user, as well as calling the function that encrypts the passwords and save the result.

int checkPermisess(thread\_data\_t\* data ,int account\_num,char \*other\_account);

Check wich users have permissions

int AddPersmisses(thread\_data\_t \* data ,int account\_num, char other\_account [])

Since we have the option to share files, this function allows us to create a folder where there is a file with the ID's of the users that have the permissions to see the files of the original user.

int DeletePermisses(thread\_data\_t\* data ,int account\_num,char other\_account[])

Just as there is the option of giving permissions, they can also be revoked, providing the ID to which the permissions will be revoked, it will be deleted from the file.

void SaveFile(thread\_data\_t\* data, int account\_num,unsigned char \* message,int numbytes, char \* fileName, char \* path, int flag)

This function is responsible for taking the files uploaded by the user and saving them on the server.

void Delete(thread\_data\_t\* data, int account\_num)

This will erase the files that the user has on the server.

char \* read\_line\_by\_line(FILE \*fin)

Returns a char pointer with the text of a file.

void printLocalIPs()

Prints the local IP.

int initServer(char \* port, int max\_queue)

Initialize the server in the given port.

int connectSocket(char \* address, char \* port)

Creates the connection to the server.

int recvString(int connection\_fd, char \* buffer, int size);

int recvFile(int connection\_fd, char \* buffer, int size);

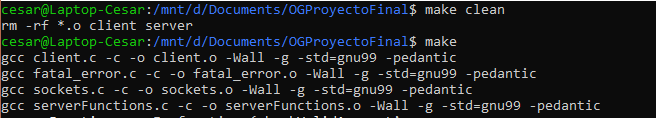
Functions that manages the buffer for communication between the server and the users.

bool saveFile(unsigned char \* file, char \* fileName, FILE \* infile, int numbytes)

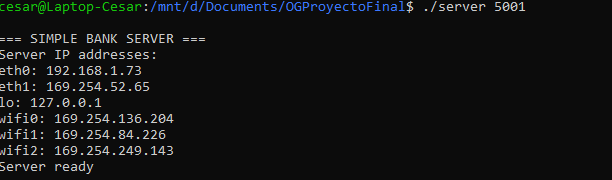
Save file in the user’s PC

**Manual:**

once we have the folder with all the files, and in the terminal we are in this folder, just to make sure we type "make clean" to delete the .o files, then we will only type "make".

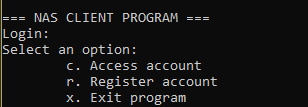


Then we use “./server <port>” where we specity the port that will be use for the server,   
and it will show a message that the server is waiting for connections:

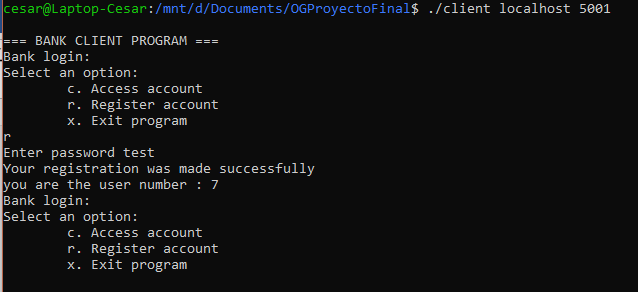


It is then when the user can already connect, in this case we will use another terminal and the address of localhost, but if it is done from another computer it is necessary to have the IP of the server.

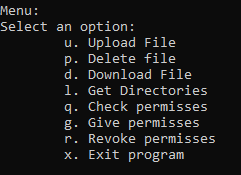
We will see the menu of the client where 3 options will be shown: log in, register and exit.



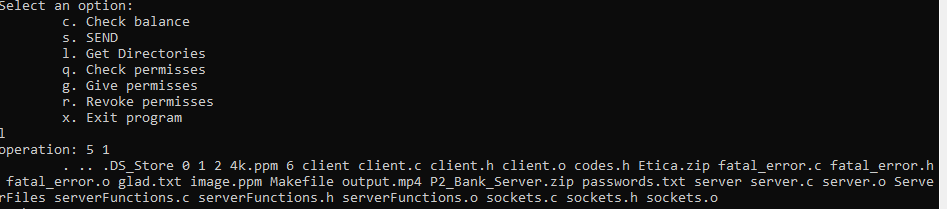
To register an account first we type "r", and then provide a password, when the account has been created we'll receive the ID to login.



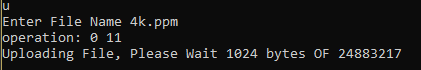
We can now login, we type "c" and provide the ID we received, in this case 7, and the password "test".



Fist with “l” we can see the files that are in the actual folder.

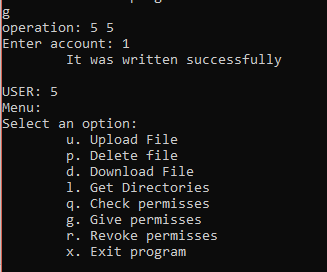


Then we have the option "u", to send the file to the server, you just have to type the name of the file, in this case, an image .ppm

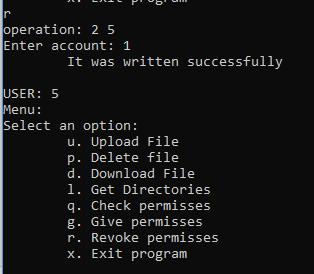


Type “d” to download an existing file and “p” to delete the file.

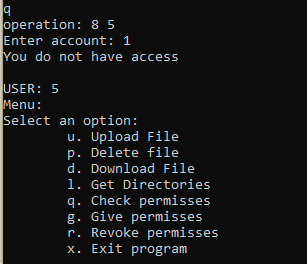
With the option “g” you give permissions to another account, you have to type the ID of the other account.



Press “r” to revoke permisses, then type the ID of the other account.



Type “q” t osee if you have permissions to see another files. it should say that it does not have permissions due it is a new account and no other user has added it.



Finally, by pressing "x" the client's account will be closed as well as the program.

