

For this project, we used two main .c files to organize client and server code. We also implemented header files to clean up the code base. We created a thread for each client, which would be handled by the server. The client calls sleep and waits 3 seconds before retrying connection to the server. The server then uses a void*/void* function, `client_handler`, which handles all commands that the client will send to the server, given the client socket as an argument. We used one mutex for the repository, locking it when the client sends a command to the server and unlocking when the command is finished running.

On the `server.c` file, besides the socket/thread creation and handling, the .c file reads in a command string given to the client in one single character stream delimited by colons. The server then calls commands based on the stream, which is then handled in `server.h`.

In `server.h`, each command that connects to the server has its own function. All of them return integers, 0 on failure and 1 on success, in order to write to `stdout` the proper messages for success or failure.

On the `WTFClient.c` file, the client first reads in the command entered. If the command requires a connection to the server, a connection is attempted and returns a socket file descriptor if successful. When a connection is successful, `WTFClient.c` will write to the server the command it read. Afterwards, `WTFClient.c` will perform further operations based on the command read and possibly read from the server for a response (if needed). Success and failure messages are always printed to `stdout`.

It is worth noting that `WTFClient.c` maintains a `clientID.txt` that holds a ID given by the server after a commit. This is used for the server to read (to distinguish different commits when a client calls `push`).

Backups were stored inside each project's directory, in a folder called `.Backups`. The backups were stored as `backup(versionnumber).tar.gz` compressed directories. History of successful commits to the project were stored in a `.History` file inside the project's directory.