Project Description

The goal of this exercise is to learn few basics about C programming by developing a Client/Server architecture using low-level TCP/IP communication.

The basic idea : You should develop a client –server system for storing and retrieving files from the server.

The work is mulitilevel and the final mark depends on which level you reach

Level 1 (basic mark 4.0) :

Develop a client and server where

1. The server starts and prints out it IP address and port to which is expecting connections
2. The client starts and asks, or receives as arguments the server IP address and port to connect
3. The client connects to the server (both print out a status that all is ok (ex. Server: got connection from 93.34.35.66 port 146, Client connected to server)
4. The client will be able to send 2 types of requests to the server
    a. List – and the server will reply with a list of available files
    b. Get <filename> - and the server will send to the client the named file
5. The server next waits for another command (list or get)
6. Last command : exit and the client and server programs terminate

Level 1.5 : (mark 4.5)

Incorporate some basic error control

- Wrong command (ex : geet (instead of Get), or print (non-existing)
- Non-existing file name

Level 1.7 (mark 4.75) – error control

- Client failed to connect to the server (wrong server IP or port number)

Level 2 : (mark 5)
Implement File uploading – from client to server

1. The client sends a file to the server - command “put <filename>”

Level 2.5 : (Mark 5.5)

- The uploaded file is renamed : Put <originalFileName> <newFileName>
- Error control :
    - file at client side not existing
    - Filename at server side already there

Level 3 : (Mark 6)

The server “fork”s a second process to handle the client just connected, and then waits for a second connection (so we can have more than one clients connected in parallel)

Level 4 : (mark 7!!!)

Instead of programming a full server, you program a light server that waits for connections and a normal server that handles the connection. Once the light server gets the connection and fork a new process, the new process “exec” the real server process, while the server waits for new connections.

UNIVERSITÉ
DE GENÈVE