IPK
2017/2018

First project:
Client-server application for obtaining user information

Jakub Crkoň, xcrkon00
Fakulta informačních technologií
VUT v Brne

Table of contents

1. Project assignment

Goal of the project was to create client-server application for obtaining user information of UNIX based operating system from /etc/passwd file using BSD sockets.

2. Project implementation

Server application has to be running before client application is executed. Communication is always started by client. Client connects to specified port on which server is listening for incoming connections.
Client requests information about user by sending a message of specific format to the server. After message is send , communication socket is closed for write on client-side , letting server know that there is nothing else to receive.

After receiving the message server verifies if it was send by the right client. Verification is performed by checking for protocol header. If header was found server proceeds with analyzing the rest of the message. Depending on the selected option, server gets needed information from the '/etc/passwd' and sends it back to client with added header '<<xcrkon00#flag>>' (flag can be capital letter S or F). When all server-side operations are done, connection with client is closed.

When client receives a response from the server it checks header which determines whether the requested user data was found **and outputs the message received from the server to standard output stream or standard error stream.**

3. Protocol information

Message accepted by server is consisted of two parts, first part is protocol header '<<xcrkon00#IPK>>'. Second part is option in the form of capital letter N, F or L followed by user's login and separated by hashtag('#').

Response accepted by client is consisted of header <<xcrkon00#S>> as success or <<xcrkon00#F>> as failure followed by user information requested.

4. Details of implementation

While gathering needed information about BSD programming and reading manuals of needed functions, I found information that gethostbyname() is outdated. Instead of it, this implementation uses getaddrinfo() for getting information about the host. Iterating over found host addresses, binding in case of server or connecting in case of client to first option available is performed.

Both client and server always send full message and then close socket (client closes socket only for writing as it will receive response from server), letting each other know that there is nothing more to receive. Then client prints received message and exits. Server is running until receiving signal SIGINT.

After being set to listening, server runs in infinite loop waiting for connections from clients. For purpose of being able to respond to more clients at once, main server process creates a child process for each connected client. Parent process then close socket for communication with client and child process close socket responsible of listening for new client connections. This way, it is possible for main server-side process to accept new client connections while child processes are communicating with clients.

5. Examples of usage

Server execution:

./ipk-server -p port

Client execution:

./ipk-client -h host -p port [-n|-f|-l] login

Real example

Provided that server application is already running.

Client request:

./ipk-client -h merlin.fit.vutbr.cz -p 55555 -n xcrkon00

Server response:

Crkon Jakub,FIT BIT 2r