Operating Systems 0512.4402

Homework 3: Chat client/server in Linux

In this exercise we will build client/server chat program in Linux using the TCP protocol. We run the server once, listening on a specified TCP port. Multiple clients can then connect to the server and chat between them. A line entered in one client is sent to the server and then sent from the server to all the clients

Server command line syntax

hw3server port

- port: server TCP port number.

Client command line syntax

hw3client addr port name

- addr: server address.
- port: server port.
- name: client name.

Server operation

The server binds a socket to the specified TCP port number and listens for incoming connections. Each time a new connection is accepted, a line should be displayed “client name connected from address”, where address should be the IP address of the client, and name should be replaced by the client name (the last command line parameter in the client invocation).

Each accepted connection creates a new socket. The server should continously monitor incoming messages on those sockets, while simultaneously accepting new connections so other clients could connect.

There are two possible messages from a client. The first is a “normal” message, which is just any text, and should be sent to all connected clients (including back to the client who sent it earlier to the server).

The second message type is a “whisper” message, which is a private chat message and should be sent just to the destination client. The whisper message syntax is “@friend msg”. It starts with the symbol @, and “friend” is the client name the message is destined for. Then “msg” is just any text.

For both the whisper and the normal messages, the server adds a prefix “sourcename: “ before sending to clients, where sourcename is the name of the client who sent that message to the server, then we have “:”, then a space, and then the original incoming message.

The server should monitor client disconnections. Each time a client disconnection is detected, a line should be displayed “client name disconnected”, where name is replaced by the client name.

Client operation

After establishing a connection to the server, the client should notify its name to the server (the details of how to do this is up to you).

Afterwards, the client displays any incoming messages from the server, as is.

In parallel to displaying incoming messages, the client should allow the user to type messages. There are three types of messages that the user can type:

- normal message: just a text line (finished with newline).
- whisper message: like a normal message, but prefixed with “@friend “ (the symbol @, then friend is the destination client name for whom the private message is for, then space, then any text and a newline.
- !exit This is handled like a normal message (sent to the server and the server will sent it to all clients), but then we print “client exiting” and exit the client.

Make sure that the client continues to display incoming messages from the server even while accepting a new input line, so that we will not delay incoming messages from other clients just because we’re accepting a new message from the user.

Submission guidelines

- The solution should be submitted in moodle in a gzipped tar file called hw3_id1_id2.tgz, where id1 and id2 are the “tehudat zehut” of the two students (or hw2_id.tgz if submitting alone).
- The tgz file should contain a subdirectory hw3_id1_id2, and in the subdirectory there should be a Makefile, as well as the .c and .h source files. Running “make” should build the code into two executables called “hw3server” and “hw3client”. Running “make clean” should clean up the executables and object files.
- Make sure to have comments in your source code.
- Submit an external documentation pdf in a file called hw3_id1_id2.pdf, describing your solution.