Task documentation for simple chat client program for IPK 2016/2017
Name and surname: Juraj Ondrej Dúbrava
Login: xdubra03

Description:

Chat client program provides basic functionality for its users to interact with other users: user can send messages to other users through chat server and receive messages from users through server. Server is used as a mediator, if it receive message from connected user, it sends that message to all connected users. Client program can be terminated after SIGINT signal from user.

Program parameters:

Client program has to be run in following format : **./chat_client -i IP_address -u username.** Program has two parameters, which must be always entered. First parameter -i IP_address defines address of chat server used for communication, second parameter -u usermane specifies username used for chat.

Implementation:

Parameters check:

At first, program provides processing of input parameters. If number of parameters is correct, IP address of server and username are stored to **chat_info structure** , which holds these values. Parameter **-i IP_address** has to be always first, else error for bad arguments occurs and program ends.

Message sending

Message sending provides function **send_message().** Function reads input entered by user from standard input in a loop and send it to the server after newline is entered.
Sent message has following format**: username: body of message\r\n**.

If EOF is read, reading in loop is terminated and user is only able to receive messages . This could happen if input is provided from pipe.

Message receiving

Message receiving provides function **receive_message().** Function receives data in an infinite loop using 1024 characters data buffer. When receiving is done correctly, message is printed to standard output.

Communication

Program uses IPv4 protocol for communication with chat server. After processing arguments, communication with server is being set up. If program cannot connect to server, an error occurs and program ends.

Communication begins with sending login message, which tells other users that certain user logged in. After that, user can send and receive unlimited number of messages. To provide sending and receiving messages at the same time, program needs to use threads for handling these tasks. Two threads are created. **send_thread** for handling message sending and **receive_thread** for handling message receiving.

Program can be anytime terminated by SIGINT signal from user. For handling this situation is used function **my_handler()**. Function first sends logout message, which tells that user ended participation in chat, close socket used for communication and exit program.