Project 1: A Broadcast Chat Application

Due: 10/05/2023

Programming Languages you can use

C/C++, Java, and Python, please see below for more details.

Project Description

A broadcast chat application has two components: a chat server and chat clients. A chat server is an application program that enables conversation between multiple chat clients. Clients connect to the server and all the connected clients receive all the messages transmitted by any of the clients. This group communication is facilitated by the chat server. A client reads the text typed by the user and sends it to the server. It is the responsibility of the server to relay that text to all the other clients. The clients receive the text relayed by the server and display it on the screen.

In this project, you need to write two programs chatserver and chatclient, which correspond to the chat server and chat client, respectively. The BSD sockets are used for communications between the client and server. The functionality of each of these programs is described below.

chatserver

usage: chatserver

The server first creates a socket using the socket() system call. It then binds its address to the socket using the bind() system call. It specifies the port number as 0, which allows the system to assign an unused port number for the server. The assigned port number and hostname are displayed on the screen. Then listen() call is used to indicate that the server is now ready to receive connect requests from clients.

The server's job is to wait for either connect/disconnect requests or messages from clients. If it receives a connect request, it accepts the connection and adds it to the list of established connections which it has to monitor for messages. If it receives a disconnect request, it closes the connection and removes the client from the list of established connections. Otherwise, it forwards the message to all other clients. This monitoring of events from several connections can be performed by using select() system call.

chatclient

usage: chatclient <servhost> <servport>

where servhost can be either the domain name or IP address of the server. servport is the port number of the server.

The client first creates a socket using the socket() system call. It then connects to the server using the connect() system call. The whereabouts of the server are passed as command line arguments.

Once the connection is established, the client's job is to wait for either user input from the keyboard or messages from the server. Inputs from the user are sent to the server and the messages from the server are displayed on the screen. Once again, select() call is used for monitoring the input from keyboard and socket.

Deliverables Requirement

Tar all the source code files and the makefile in a single tar file and submit on Canvas. Here is a simple example on how to use the Unix command tar (assuming all your files related to this projects are under one directory, say project1). To tar all the files:

tar -cvf proj1.tar *

To check the contents in the tar file:

tar -tvf proj1.tar

To untar a tar to get all files in the tar file:

tar -xvf proj1.tar

At the beginning of all your source code files, you must include the following information:

Name: (Your name)

Sample Executable Code

You can download the sample executable code of the project. The tar file contains two executable files chatserver and chatclient compiled on linprog. NOTE that, the format of messages exchanged for the provided sample code is:

long integer indicating the number of characters in the user input, followed by the user input

You may use the same message format, or your own format.

Notes

- Test your programs thoroughly. For example, when the chat server terminates (or is killed), all chat clients should be terminated. In addition, a chat client may be terminated, and then re-started.
- We must be able to (compile and) run your program on linprog. If you have any specific ways to compile and run your project, please provide a readme file.
- You can develop the project using C/C++, Java, or Python (the provided examples were written in C/C++). The way to run your programs should be similar to the above examples. You should use the common filename suffix for your programs, such as .h, .c, .cpp, .java, and .py, depending on the programming languages that you choose.