Computer Networks Lab (CS 353): Lab 1

This assignment will not be graded.

In this assignment you are going to revise the basics of UNIX socket programming. You will implement a simple client-server model where a client program can chat with a server. The communication between client and server goes as follows:

- The server gets started first and listens to a known port.
- The client program is started. The server IP and the port are provided in the command line.
- The client connects to the server and asks for user input. The user sends the Unix command for which it needs a short explanation. The input is sent to the server through a connected socket.
- The server receives the input from the client socket, and sends the result back to the client.
- The client should display the result from the server to the user and prompts for the next input until the user terminates the client program by Ctrl+C.

You are provided with the Client skeleton source code. Write C/C++ code for **two** versions of the server.

1. Your server program 'Server1' is a single process server that can handle only one client at a time. If another client tries to chat with the server while one client session is going on the second client will receive an error message like "line busy!"

2. Your server program "Server2" is a multi-process server that forks a process whenever it receives a new client request. Multiple clients will be able to chat with the server concurrently.

Use "man <command>" to get an explanation of what the command does.

You may use Java instead of C/C++.

Example:

The Client code is first compiled. The server runs on port 5555 in another terminal. The client program is then given the IP (localhost 127.0.0.1 in this

case) and port (5555) as command line inputs. Sample runs of the client goes as follows.

$ gcc -Wall client.c -o client

$ ./client 127.0.0.1 5555

Connected to server

Please enter the message to the server: ls

Server reply: ls – list directory contents

Please enter the message to the server: cat

Server reply: cat – concatenate files and print on the standard output

The sample run of the server is as follows.

$ gcc –Wall server1.c -o server

$ ./server 5555

Connected with client socket number 4

Client socket 4 sent message: ls

Sending reply: ls – list directory contents