Recommended Programming Assignment 0 (MP0)

No due date. This MP0 will not be graded. No credit assigned to MP0.

- If you are not familiar with socket programing: This assignment is strongly recommended for students who do not have socket programming experience. The purpose of this assignment is to familiarize you with socket programming. Students who are not familiar with socket programming are encouraged to complete MP0 early in the semester.

- If you are familiar with socket programming, we recommend doing the last part (message-passing) of the assignment below as a refresher.

For this assignment, you may want to start with `client.c`, `server.c`, `talker.c`, and `listener.c` source code available from Beej's Guide to Network Programming, available at:

http://beej.us/guide/bgnet/

The instructions below assume that you use the above code. However, if you decide to use another language, such as Java, you may find similar code available elsewhere (for instance, an older edition of the networking textbook by Kurose and Ross contains Java code).

You should try to understand what these programs are supposed to accomplish. Reading Beej's guide can be helpful. You may also find useful information in many books (e.g., the CS/ECE 438 textbook by Kurose and Ross, or UNIX Network Programming by Stevens).

Execute `client` on one and `server` on different machines if available. Alternatively, you may run them on the same machine, but use different port numbers for the two programs. They form a socket connection over TCP.

Similarly, execute `talker` on one machine and `listener` on different machines (or same machine). They communicate using UDP.

Next, change `server.c` to accept a file name as a command line argument and to deliver the length and contents of the file to each client. Assume that the file contains no more than 100 bytes of data (assume that it is a text file, and ignore additional data). Send the length of the file (an integer between 0 and 100) as a 32-bit integer[1], followed by the data bytes. Change the client to read first the length and then the amount of data specified by the length from its TCP socket. You may want to use the `read` and `write` system calls in place of Beej's calls to `send` and `recv`.

The client output should look like this:

```
client: connecting to hostname
client: received filelen bytes

This is a sample file that is
sent over a TCP connection.
```

Where `hostname` is the address of the server, `filelen` is the number of bytes received, and the rest of the output is the file content.

Message-passing: Define a message format of your own choice (e.g., a message may represent an e-mail, with a To field, a From field, a Date field, and a plain-text message). Send information in the chosen message format over the two different types of sockets.

[1] in network byte order