ECE435: Network Engineering – Homework 1
Sockets Programming

Due: Friday, 31 January 2025, 11:00am

This Homework is meant to get you started with socket programming. It should run on any Linux/UNIX/MacOS machine. If you do not have access to such a system let me know and I can provide access.

1. **Download and Build the Code**

   (a) Download the code from:
   `https://web.eece.maine.edu/~vweaver/classes/ece435/ece435_hw1_code.tar.gz`

   (b) Unpack the files:
   `tar -xzvf ece435_hw1_code.tar.gz`

   (c) Build the C files:
   ```
   cd ece435_hw1_code
   make
   ```

2. **Test the Code**

   (a) It might be easier to see what's going on if you have two terminal windows open.

   (b) In one, first run `./server`

   (c) In another, run `./client`

   (d) Type a message on client, and it should travel over the network to server and appear on the server.

   (e) Take a look at the code and see how it works.

   (f) Simple socket code can often get an error on restart such as `Error binding! Address already in use` after exiting the program suddenly without properly closing the network connection. This is because the connection enters the `TIME_WAIT` state which lasts 60 seconds as the OS waits in case any lingering packets are still on their way. We include a `setsockopt()` call to avoid this.

3. **Modify the server to echo back data from client (3pts total)**

   (a) **Modify the server to echo**
   First modify the server code (`server.c`) so that it gets the message from the client and sends the same message back.
   Just send back the received message back, no additional text.

   (b) **Modify the server to not exit**
   Modify the server code (`server.c`) so that instead of exiting after one transaction, it instead loops forever reading from the file descriptor and responding

   (c) **Modify the client so that it does not exit**
   Modify the client code (`client.c`) so it loops forever, waiting for a message to be typed then sending it. You can always use control-C to quit.

4. **Modify the code to exit on command (2pts total)**

   (a) **Server closes on command** Modify the server code so that if the string `bye` is received, it exits the server.
   You can use the `strncmp()` function for this, but beware the unusual behavior of `strncmp()` (0 means a match)
   Also note that `fgets()` is going to leave the trailing linefeed at the end of the string so take that into account.

   (b) **Quit client on exit** Once "bye" is echoed back from the server, detect this on the client and exit the client too.

5. **Modify the server to uppercase the text (1pt total)**

   (a) **Have the server uppercase the string** Modify the server so that when it receives the string it converts all of the lowercase characters to uppercase before sending back the uppercased response.
   You might find the `toupper()` function useful. Be sure the client still exits properly on "bye" after this change.

6. **General Correctness: (1pt)**

   - Be sure to comment your code!
   - Be sure to fix any warnings that the compiler gives.
   - Be sure to handle the case where the client disconnects unexpectedly (for example, control-C pressed before any data sent)
   - Be sure to handle buffer overflows.
     What if the client sends 100kB of raw data to your server?
     The server should not crash in this case.
   - Be sure you are keeping the same connection open and *not* starting a new connection for each string. (i.e. Make sure you are looping to the right place in the server). This is tricky because in the server there are two different file descriptors involved.
   - Your client should keep reading from the server until the full response comes back. (i.e. if you type "hi" and press enter, you should get the "HI" response back without having to type anything else).
   - Your client and server should only send data you intend to send. So if you only have 5 bytes of data, be sure to send that and not possibly a full 100 byte buffer than starts with your 5 bytes followed by lots of 0s or other values. The usual cause of this is confusion of `strlen()` with `sizeof()`. To measure the length of a C string use `strlen()`, sizeof on an array is probably going to return the full size of the buffer array, it doesn't know or care about what's inside of it.

7. **Something Cool (1pt)**

   Before you start this, copy your working `server.c` and `client.c` files on top of `server_cool.c` and `client_cool.c`. This makes things easier to grade by separating out the cool work. To do this from the command line you can do something like:
   `cp client.c client_cool.c`

   You can do one of the following:

   - Modify the server to get the port number from the command line (look into `atoi()` or `strtod()`).
   - Modify the client to get both the hostname and port from the command line. The code should default to the original values if no options are specified.
   - When a connection comes into the server, print the port and address of the incoming client. This info can be found in the `client_addr` structure.
   - Modify the client or server to use colors when printing messages and the text.
     HINT: Look up "ANSI escape codes"
     Note: don't send the escape codes over the socket connection, just print the colors locally.

8. **Answer the following questions** (2pts total)

   Short answers are fine. Put your answers in the `README` file using a text editor, it will be automatically included in the submission process.

   (a) In the OSI reference model, which layer deals with routing packets from one network to another?

   (b) In the OSI reference model, which layer deals with the actual bits, voltages and frequencies involved (give the name, not just the number)?

9. **Submit your work**

   - Please edit the README file to include your name.
     Also put your answers to the questions there.
   - Run `make submit` which will create a `hw1_submit.tar.gz` file containing `README`, `Makefile`, `server.c` and `client.c`.
     You can verify the contents with `tar -tzvf hw1_submit.tar.gz`
   - e-mail the `hw1_submit.tar.gz` file to me (vincent.weaver@maine.edu) by the homework deadline. Be sure to send the proper file!