1 Description

In this assignment you will develop a TCP server. We will provide you a TCP client that you may use to test your server.

2 Functionality

This section describes the functionality that your server needs to support:

1. The client we provide you is invoked
   ruby client.rb --ip <SERVER_IP> --port <SERVER_PORT> --num <NUM_BYTES>
   - SERVER_IP: IP address of your server.
   - SERVER_PORT: Port your server listens to.
   - NUM_BYTES: Number of bytes that clients send to the server.

2. The client will produce a random string of length NUM_BYTES and will send it to your server.

3. The string contains lower case characters between [a - z].

4. The client terminates every message it sends with a special character sequence \n\n\r; therefore the size of the message sent is NUM_BYTES plus 3.

5. Your server should receive the whole message, convert its characters to uppercase and print it on the standard output.

6. Take advantage of the spacial trailing character sequence in order to know when to stop waiting for more data from the client.

7. Make sure that your read all the data the client sends. To ensure this, read from the socket inside a while loop and keep track how much you read each time. Read the man page of the socket recv(2). Stop reading from the socket only when you have seen the trailing character sequence. The behavior of recv(2) does not guarantee that you will get all the data that the clients sends with only a single call of the function.

8. To terminate your server add a signal handler that catches the SIGINT signal; this the signal when you press ctrl-c on your keyboard.
   (a) Register a signal handler; use signal(2). The first argument of signal(2) is the signal you want to catch. The second argument is a function that it will be executed when the operating system sends the signal of interest (SIGINT in this case).
(b) In the definition of your signal callback function make sure you close the socket of your server (use `close(2)`).

9. Your server should take two command line arguments: an IP address and a port.

2.1 Remarks

1. You must use the C programming language.

2. You must submit code that compiles, otherwise your assignment will not be graded.

3. Your code must be `-Wall` clean on linux-lab’s gcc otherwise your assignment will not be graded.

4. You must submit a tar.gz file that contains a directory with your solution.

5. The name of the directory that includes your solution should follow the format: `<username>_assignment1`. The username should be the same as it appears on the grades server (grades.cs.umd.edu).

6. The filename of your solution should be `<username>_server.c`.

7. You must provide a Makefile.

8. You are not allowed to work in teams or to copy code from any source.