1 Introduction

In this assignment you will write the server program which will communicate using sockets with the client program in the previous assignment.

2 The Protocol

Your server will run on a linuxlab machine and will listen on a TCP socket bound to a port as described. Note that you cannot bind to a port below 1024 without having super user (root) access.

Given that your class account login id is cs4170xx, the ports you should use are 10xx0-10xx9 (inclusive). Thus, if your login id is cs417060, you will use port range 10600-10609. Your final project must work with any port, but when you are testing your project you should only use the ports you have been allocated to avoid collision with others. The rest of the protocol is as described in assignment 1.

3 The server program

The command line syntax for a minimal server is given below. The server will take the port as an argument.

server [<port>]

• The cookie should be generated using the formula:
(a + b + c + d) × 13 mod 1111, where a.b.c.d is the IP address of the client, 0 ≤ a, b, c, d ≤ 255.

• After successful communication, the server MUST print the cookie it generates along with the client's login id, first name, IP address and port number. All this information should be in a single line. An example is:

555 cs417050 Alice from 128.8.126.208:48542

Notice that ((128+8+126+208) × 13 mod 1111) is 555

• Your server should not accept spurious input from the clients.
  • We will test your server with non-conforming clients; the server should print out an error message containing the client's IP address and port number also in a single line, as such:

  **Error** from 128.8.126.133:48522

  and immediately close the connection when it finds a bad message from the client. It should not breakdown, but continue operating after servicing misbehaving clients. Bad messages, as per assignment 1, are ones that have an incorrect magic string, incorrect message type or too many fields.
Remember, the cookie sent in the STATUS message has to match the cookie in the CLIENT BYE message for a communication to be successful.

The server should be able to serve multiple clients. It is acceptable if this is done serially.

• Do NOT print out any other debugging messages. They are useful for you, but not for your TA to grade.
• All output should be printed to stdout. Use `fflush(stdout);` after every output to stdout.

4 Requirements

• We provide a conforming client for those who did not get the client to work and a public test to test the server functions against it.

• Your code must be -Wall clean on gcc.

• The TA will answer general questions/confusions only, and is not supposed to debug for you. The “This is my code, what could be the problem” type of questions will be ignored.

5 Project Submission

• Please submit your code to the Submit Server (https://submit.cs.umd.edu/).

• You should upload a zip file which contains the files `server.c` and `common.h` (and possibly any other .h files you are using). You can create this file on linuxlab using:

```
zip server.zip server.c common.h
```