Programming Assignment 1

Due: 09/10/2009

You should use the CSIC Linux cluster systems

1 Introduction

For this assignment, you will write a client program which will communicate using sockets with a server program provided by us. We will give you a sketch of the client program — all you have to do is fill in socket-specific bits.

For each client request, our server generates and returns a random number, and then the server and client embark on a long goodbye. Obviously, the protocol is trivial/useless, however, this exercise will get you started on the cluster and familiarize you with sockets, network programming and distributed debugging.

2 The Protocol

The server runs on the machine SERVER HOSTNAME and listens for requests on a TCP socket bound to port SERVER PORT. Both constants are defined in the header file provided and you. This exercise has four types of messages: HELLO, STATUS, CLIENT BYE and SERVER BYE. Each message is an ASCII string, and consists of multiple fields separated by whitespace (space (0x20) or newline (0x0a) character). The MAXIMUM length of the string is 255.

![Client Server Diagram]

Figure 1
The protocol outline is given in Fig 1. The client initiates the protocol by sending a HELLO message to the server. The server replies with a STATUS message. The client then sends a CLIENT BYE message, and the server terminates the connection by sending a SERVER BYE message. A connection is successful if and only if all of these messages are correctly sent and received. Since we are using TCP for communication in this assignment, you do not have to worry about lost messages etc.; you only need to ensure that all messages are sent correctly (and that you receive and parse messages correctly).

The details of each message are as follows:

- **HELLO (From the client to the server: Client → Server)**
The HELLO message has 4 fields EXACTLY in the following order
  - Magic String
    It MUST set to be MAGIC STRING which is a constant defined in the header file (cmsc417fall09). If you send a message which does not start with this magic string, the message will be ignored.
  - Message Type
    The type string MUST be HELLO to indicate a message type HELLO. The server is case-sensitive.
  - Login ID
  - Name
  The last field is your first name. Please do NOT put spaces in your name, even if it contain spaces.

  An example HELLO message might look like this:
  cmsc417fall2008 HELLO cs417000 Alice

- **STATUS (Server → Client)**
The STATUS message has 4 fields in the following order:
  - Magic String
    Same as above.
  - Message Type
    Must be set to STATUS.
  - Cookie
    An integer randomly generated by the server (represented in ASCII).
  - IP Address and Port number
    A string of the form a.b.c.d:e, representing the IP address and port number of the client.

  An example STATUS message might be:
  cmsc417fall2009 STATUS 42 128.8.128.153:392932
• CLIENT BYE (Client → Server)
The CLIENT BYE message has 3 fields in the following order:
– Magic String
   The same as above.
– Message Type
   Must be set to "CLIENT_BYE".
– Cookie
   A string of an integer, set to the value of the cookie sent by the server in the STATUS message for this connection.

An example CLIENT BYE message would be:
cmsc417fall2009 CLIENT_BYE 42

• SERVER BYE (Server → Client)
The SERVER BYE message has 2 fields in the following order:
– Magic String
   The same as above.
– Message Type
   Must be set to "SERVER_BYE".

An example SERVER BYE message would be:
cmsc417fall2009 SERVER_BYE

3 The Client Program
The command line syntax for the client is given below. The client program takes command line arguments corresponding to the login id and first name. The hostname and port specifications are optional. If included, they override the default definition of SERVER HOSTNAME and SERVER PORT in client.h.
```
client [<hostname>[ <port>]] <login id> <first name>
```

4 Requirements
You may test your client code with our server as many times as you like. You will be building on these programs for subsequent stages of the term project so it is in your own best interest to make them maintainable. Your client program must verify the validity of messages by checking the magic string and message type fields in STATUS message. If a received message is not as expected, such as an incorrect magic string or wrong message type, assert an error and terminate your program.

Your code must be -Wall clean on gcc. Do not ask the TA for help on (or post to the newsgroup) code that is not -Wall clean unless getting rid of the warning is what the problem is in the first place.