

Programming Assignment 0

Assigned: 01/30/2009

Due: 02/04/2009, 11:59:59 PM.

You should use the CSIC Linux cluster systems and obtain your account information from the TA.

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 two cookies, 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 for you. This exercise has four types of messages: `HELLO`, `STATUS`, `BYE` and `CONFIRM_BYE`. Each message is an ASCII string, and consists of multiple fields separated by spaces (0x20). The maximum length of each message is `MAX_STR_SIZE`, which is also defined in the header file given to you.

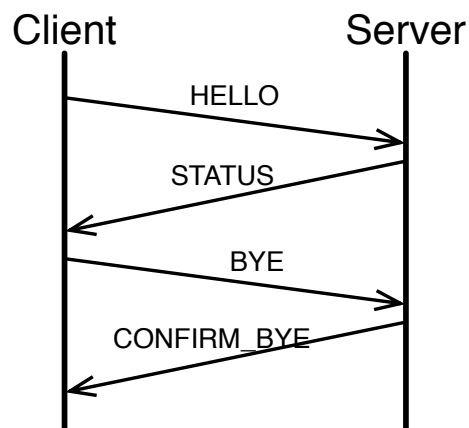


Figure 1: protocol outline

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 `BYE` message, and the server terminates the connection by sending a `CONFIRM_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

- **Message Type**

The type string MUST be HELLO to indicate a message type HELLO. The server is case-sensitive.

- **Magic String**

It MUST set to be MAGIC_STRING which is a constant defined in the header file (`cmisc417spring09`). If you send a message which does not contain this magic string, the message will be ignored.

- **Login ID**

This field is your cluster login ID.

- **Last Name**

The last field is your last name. Please do NOT put spaces in your last name, even if it contain spaces.

An example HELLO message might look like this:

```
HELLO cmisc417spring09 cs417000 Huang
```

- **STATUS** (Server → Client)

The STATUS message has 5 fields in the following order:

- **Message Type**

Must be set to STATUS.

- **Magic String**

Same as above.

- **Cookie1**

An integer randomly generated by the server (represented in ASCII). The range is between 1 and 1000.

- **Cookie2**

Another integer randomly generated by the server (represented in ASCII). The range is between 1 and 1000.

- **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:

```
STATUS cmisc417spring09 356 478 128.8.128.153:39293
```

- **BYE** (Client → Server)

The BYE message has 4 fields in the following order:

- **Message Type**
Must be set to "BYE".
- **Magic String**
The same as above.
- **Sum of Cookies**
An integer set to the sum of the two cookies that are received from the server (represented in ASCII).
- **Comment**
Optional string field you may send to the server. You may put any visible ASCII characters in the comment, as long as the length of the entire message does not exceed MAX_STR_SIZE.

An example BYE message would be:

```
BYE cmsc417spring09 834 See you later!
```

- **CONFIRM_BYE** (Server → Client)

The CONFIRM_BYE message has 2 fields in the following order:

- **Message Type**
Must be set to "CONFIRM_BYE".
- **Magic String**
The same as above.

An example CONFIRM_BYE message would be:

```
CONFIRM_BYE cmsc417spring09
```

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 last name. The hostname and port specifications are optional. If included, they override the default definition of `SERVER_HOSTNAME` and `SERVER_PORT` in `common.h`.

```
client [<hostname>[ <port>]] <login id> <last name>
```

4 Requirements

You may test your client code with our server as many times as you like. Your client should conform to the protocol described above, or otherwise the server will terminate the connection silently. 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` and `CONFIRM_BYE` messages. 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 forum) code that is not `-Wall` clean unless getting rid of the warning is what the problem is in the first place.