Program #1

Build a network client/server system that implements the "Knock Knock protocol" as described in this document. The client and server will communicate on TCP port 2983.

The server will listen for joke requests. The client sends message 000 to request a joke and initiate the conversation. Each message is formatted as a 3 byte code and 64 bytes of text. The entire message is a C-string, all bytes are of type `char`. Both the server and client will print out all messages sent and received. The server will only handle one connection at a time (i.e. it does not fork and is not threaded).

```
+-------------+-----------------------------------------------------+
| 3 byte code | 64 byte message                                     |
+-------------+-----------------------------------------------------+
```

The protocol codes and message strings are listed below in Table 1. Notice that the server must match the 400 code to the corresponding 300 code (the server must maintain joke state). Figure 1 depicts the exchange of messages between the client and server for the "doughnut" joke.

| Code | Message text |
|---|---|
| 000 | Tell me a joke |
| 100 | knock knock |
| 200 | who's there |
| 300 | doughnut |
| 300 | little old lady |
| 300 | who |
| 300 | Dewey |
| 400 | "300string" who |
| 500 | Doughnut ask, it is a secret |
| 500 | I didn't know you could yodel |
| 500 | Are you an owl |
| 500 | Dewey we have to keep telling silly jokes |
| 600 | Errors aren't funny |

Table 1: Knock Knock protocol

Codes by role:
Server: 100, 300, 500, 600
Client: 000, 200, 400, 600

If an error occurs in the protocol, a code 600 message is sent and the connection is closed. Errors include an invalid code, an unexpected code, or malformed message.

You may wish to model your server and client as state machines. For example, Figure 2 depicts the states and transitions for the server.

Your programs (client and server) must be developed in C or C++ using the BSD socket API on either Linux or BSD. The programs must compile and run on the class server (BSD) with gcc or g++. The class server (cs-srv-01) is accessible from the campus network and campus wireless.

**Due dates**:

Completed server - 03/04/15
Completed client - 03/09/15

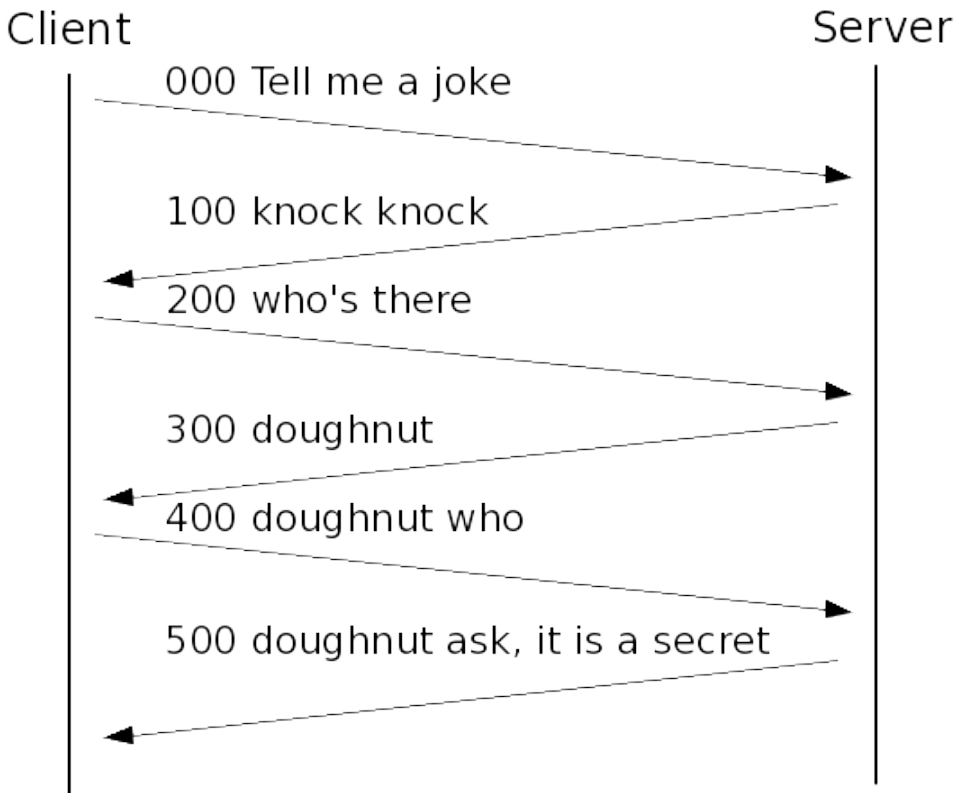


Figure 1: Client server exchange

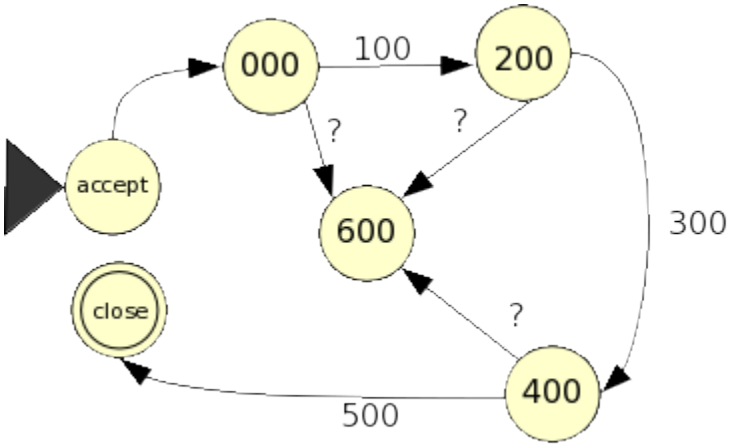


Figure 2: Server state machine

**Program evaluation**:

See the evaluation sheet for programming assignments on the course web site. Your program must meet all of these criteria in order to receive full credit. Your programs (client and server) should be submitted in source form in either a single zip, tar, or gzipped tar file. Your submission must include *all* files (headers, source files, makefile, etc.) required to successfully build your programs. I will enable all warnings and diagnostics and expect your code to compile cleanly without warnings.

Create and include a Makefile with your submission. See make(1) for details.

**Hints**:

Test your server by using telnet(1) or nc(1) as a client.

Testing your server and client on the same machine is possible (and easier). Use multiple terminal windows or a virtual terminal emulator to run your server and clients on separate terminals. tmux(1) is a terminal multiplexer that allows you to have multiple terminals in a single screen. Use localhost or 127.0.0.1 as the host. Don't forget to check your program over a network connection.

tcpdump(8) and gdb(1) are useful tools to track down unexpected problems.

**Additional resources**:

Sample code - Textbook pages 33-36
man pages - socket(2), accept(2), bind(2), listen(2), send(2), recv(2)
man pages - signal(3), getaddrinfo(3)
man pages - select(2), tmux(1), telnet(1), nc(1), tcpdump(1), gdb(1)
TCP - RFC 793, RFC 1122
Domain names - RFC 1035
`http://beej.us/guide/bgnet/`