Computer Science 367

Program 1

Read all of the instructions. Late work will not be accepted.

Overview

For this warm-up network programming assignment you will create a client and server for a simple guessing game. When the server is started, it is given a "secret" number (which is a 32-bit unsigned integer). The client must guess this number. After every guess by the client, the server reports whether the guess was too high, too low, or the correct answer. The interaction continues until the client guesses correctly. An example session, from the client's perspective, is as follows:

```
$ ./prog1_client 127.0.0.1 36799
Enter guess: 50
Too low.
Enter guess: 150
Too high.
Enter guess: 100
Too high.
Enter guess: 99
Too high.
Enter guess: 80
Too high.
Enter guess: 70
Too low.
Enter guess: 75
You Win!
$
```

For this exercise, you will be provided example code for both the server and client, in the form of demo_sever.c and demo_client.c. You are responsible for implementing the communication protocol between the server and client to this example code. Your server and client will be tested on the department's Linux pool.

Program 1 Specifications

Your program must be compliant with all of the following specifications in order to be considered correct.

File Naming Requirements

Spacing, spelling and capitalization matter.

- Your client source code should be contained in a file named `prog1_client.c`. If you create a corresponding header file you must name it `prog1_client.h`.
- Your server source code should be contained in a file named `prog1_server.c`. If you create a corresponding header file you must name it `prog1_server.h`.

Command-Line Specification

The server should take exactly two command-line arguments:

1. The port on which the server will run, a 16-bit unsigned integer
2. The number that the client must guess, a 32-bit unsigned integer

An example command to start the server is:

```
./prog1_server 36799 55
```

The client should take exactly two command-line arguments:

1. The address of the server (e.g. a 32 bit integer in dotted-decimal notation)
2. The port on which the server is running, a 16-bit unsigned integer

An example command to run the client is:

```
./prog1_client 127.0.0.1 36799
```

Compilation

Your code should be able to compile cleanly with the following commands in the linux pool.

```
gcc -o prog1_server prog1_server.c
gcc -o prog1_client prog1_client.c
```

Protocol Specification

The communication protocol between the client and server is summarized by the following rules:

- No information is exchanged until the client makes its first guess
- A client makes a guess by sending a 32 bit unsigned integer in network byte order
- After sending a guess, the client awaits a response from the server
- Upon receiving a guess from a client, the server checks to see if the guess higher or lower than its secret number:
  - If the guess is correct, the server sends back the character '0'. The client should print "You Win!\n". After sending this message, the server closes the socket that it was using to communicate with the client.
  - If the guess is too low, the server sends back the character '-1'. The client should print "Too low.\n".

  - If the guess is too high, the server sends back the character ‘1’. The client should print “Too high.\n”.
  - As long as the message is *not* ‘0’ (denoting a correct guess) the client should prompt the user for another guess, and repeat the process.
  - Once the client receives the message ‘0’ it closes its socket and exits.

- The server is required to be capable of handling multiple clients simultaneously, each client’s guessing game is treated separately, with no interaction between clients. This will be accomplished using fork()

Submitting your work

Submit your work via a .zip or .tar file to canvas. Be sure to include:

- `prog1_client.c` and `prog1_server.c`
- Optionally, the header files `prog1_client.h` and `prog1_server.h`

Your submission need not and **should not contain your compiled binaries / object files.** I will compile both of your programs, run them in a series of test cases, and review your code.

Academic Honesty

To remind you: you must not share code with anyone other than your professor: you must not look at any one else’s code or show anyone else your code. You cannot take, in part or in whole, any code from any outside source, including the internet, nor can you post your code to it. If you need help from any other groups, all involved parties *must* step away from the computer and *discuss* strategies and approaches - never code specifics. I am available for help during office hours. I am also available via email (do not wait until the last minute to email). If you participate in academic dishonesty, you will fail the course.