Computer Science 367

(Pair) Program 0 (15 points)
Due Wednesday, Jan 21, 2015 at 10:00 PM

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

Overview

For this warm-up network programming assignment you and your assigned partner 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 warmer (closer to the solution), colder (further from the solution), the same distance to the solution, or the correct answer. The interaction continues until the client guesses correctly. An example session, from the client's perspective, is as follows:

```
$ ./prog0_client 127.0.0.1 36799
Enter guess: 50
Warmer
Enter guess: 150
Colder
Enter guess: 100
Warmer
Enter guess: 99
Warmer
Enter guess: 80
Warmer
Enter guess: 70
Same
Enter guess: 75
You Win!
$
```

You and your partner are responsible for implementing (in the C programming language) both the client and the server for this simple guessing game, using sockets. It will be developed in either your or your partner's Subversion version control repository. You will work together using *pair programming*, described in the next section.

Pair Programming[1]

Pair programming is a software development technique where two programmers work together in front of one keyboard. One partner types code while the other is suggesting and/or reviewing every line of code as it is being typed. The person typing is called the driver. The person

[1] These guidelines are based on a previous version developed by Perry Fizzano.

reviewing and/or suggesting code is called the navigator. The two programmers should switch roles frequently (e.g. every 20 minutes). For this to be a successful technique the team needs to start with a good program design so they are on the same page when it is finally time to start typing on the computer. **No designing or programming is to be done without both partners present!** Pair programming has been shown to increase productivity in industry and may well increase yours, but there are additional reasons it is being used in this class. First, it is a means to increase collaboration, which is something department graduates now working in industry report that they wish they had more experience with. Second, working in pairs is a good teaching tool. Inevitably, in each pairing the partners will have different styles and abilities (for instance, one person may be better at seeing the big picture while the other is better at finding detailed bugs or one person might like to code on paper first while the other likes to type it in and try it out). Because of that you will have to learn to adjust to another person's style and ideally you will meet each other half way when there are differences in approach. It's important that each person completely understands the program and so both parties need to be assertive. Be sure to explain your ideas carefully and ask questions when you are confused. Also it is crucial that you be patient! There is plenty of time allotted to complete this assignment as long as you proceed at a steady pace. Ask for help from me or the department tutors if you need it.

You will be assigned a partner by me via Canvas groups. Because there is no lab, you will need coordinate with your partner to find times when you can both be present. You will need to contact me ASAP if there is any reason you will not be able to collaborate. **Let me stress again that no designing or programming is to be done without both partners present! If I determine this happened I will fail you and your partner for this assignment.**

Program 0 Specifications

Your program must be compliant with all of the following specifications in order to be considered correct. Non-compliance will result in penalties.

File and Directory Naming Requirements

Pick one of the two partners to host `prog0` in their repository. The other partner should *not* add any `prog0` directory. The following instructions apply to the partner hosting `prog0`. Spacing, spelling and capitalization matter.

- The writeup and all of your source code should be found directly in `yourWorkingCopy/prog0`, where `yourWorkingCopy` is replaced with the full path of your working copy.
- Your client source code should be contained in a file named `prog0_client.c`. If you create a corresponding header file you must name it `prog0_client.h`.
- Your server source code should be contained in a file named `prog0_server.c`. If you create a corresponding header file you must name it `prog0_server.h`.
- Your writeup must be a plain-text file named `writeup.txt`.

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:

```
./prog0_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:

```
./prog0_client 127.0.0.1 36799
```

Protocol Specification

The protocol for this program is very simple, and is summarized by the following rules:

- Once the client has connected to the server, the server forks off a process to handle the guess game interaction, the new process initializes the "previous guess" to `0` and 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 is closer to the number than the previous guess was:
  - 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 closer, the server sends back the character '1'. The client should print "Warmer\n".
  - If the guess is further, the server sends back the character '2'. The client should print "Colder\n".
  - If the guess is the same distance as the previous guess, the server sends back the character '3'. The client should print "Same\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.

The Repository

- All code for this program will be developed under a Subversion repository. Exactly once at the beginning of the quarter you will need to create your repository and checkout a working copy to use. I will provide a script to automate this process.
- As noted above, your work will be done in a `prog0` sub-directory of your working copy. Therefore, exactly once at the beginning of working on Program 0, either you or your partner will need to create this directory and add it to Subversion. That person will also need to add your writeup and every source code file to your repository. **If you do not add and commit your files, I cannot see them, and thus cannot give you any points for them.** I can provide a script to help you determine if all of your files have been successfully added and committed.
- You must actively use Subversion during the development of your program. If you do not have at least 3-5 commits for Program 0, points will be deducted.
- See the Subversion Guide on Canvas for all sorts of details on Subversion.
- *Note: you must never edit files in your repository directly. All interaction will be indirect, via your working copy.*

Grading

Submitting your work

When the clock strikes 10 PM on the due date, a script will automatically check out the latest committed version of your assignment. I will grade your or your partners files, whoever has a `prog0` directory. **(Do not forget to add and commit the work you want submitted before the due date!)** The repository should have in it, at the least:

- `prog0_client.c` and `prog0_server.c`
- Your write-up: `writeup.txt`
- Optionally, the header files `prog0_client.h` and `prog0_server.h`

Your repository need not and **should not contain your compiled binaries / object files.** Upon checking out your files, I will compile both of your programs, run them in a series of test cases, analyze your code, and read your documentation.

Points

By default, you will begin with 15 points. Issues with your code or submission will cause you to lose points, including (but not limited to) the following issues:

- Lose a few points:
  - Not including a writeup or providing a overly brief writeup
  - Poor code style (inconsistent formatting, incomprehensible naming schemes, etc.)
  - Files and/or directories that are misnamed
  - Insufficient versioning in Subversion

- Lose a lot of points:
  - Not including one of the required source code (`.c`) files
  - Server cannot handle multiple clients concurrently
  - Code that does not compile
  - Code that generates runtime errors
  - A client or server that does not take the correct arguments
  - A client or server that does not follow the specified protocol

Write-Up

You need to create, add and commit a *plaintext*[2] document named `writeup.txt`. In it, you should include the following (numbered) sections.

1. Your name and your partner's name
2. Declare/discuss any aspects of your client or server code that are not working. What are your intuitions about why things are not working? What issues you already tried and ruled out? Given more time, what would you try next? Detailed answers here are critical to getting partial credit for malfunctioning programs.
3. In a few sentences, describe how you tested that your code was working.
4. What was the most challenging aspect of this assignment, and why?
5. What variant/extension of this assignment would you like to try (e.g. a variant that is more powerful, more interesting, etc.)

Academic Honesty

To remind you: you must not share code with anyone other than your assigned partner and 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.

[2]E.g. created with vim, kate or gedit.