CS 447 : Networks and Data Communications

Programming Assignment #01

Total Points: 150

Assigned Date : Wednesday, January 31, 2018
Due Date : Wednesday, February 14, 2018 @ 11:59:59 a.m.

Overview

Your first programming assignment is to **implement a basic client/server application** using the socket interface. There are several objectives of this assignment. These are:

a. to get your feet wet on socket programming basics;
b. to understand the ordering of the socket interface primitives;
c. to get you exposed to Linux system calls (if you already haven't);
d. to gain a basic understanding of network protocols; and
e. to set yourself up for the rest of the course.

Back Story

Captain Haddock, during his many sea voyages, likes to be able to do geometric calculations without banging his head against the ship's hull; consuming absurd amounts of Rum coupled with sea sickness has proven not be a good recipe for concentration for Haddock lately. *Professor Calculus*, who recently took CS447 at SIUE, thinks he has a solution. Calculus agreed to create an online geometric calculator application for haddock that supports both **reliable** and **unreliable** connections. Not only that, Calculus agrees to create this calculator in a way that **more than one person** can use it at the same time. Calculus plans to create the first build with support for 6 basic calculations on three geometric shapes as follows:

1. **`CIRCLE`**
   - Given radius r, calculate area A
   - Given area A, calculate circumference C
2. **`SPHERE`**
   - Given radius r, calculate volume V
   - Given surface area A , calculate radius r
3. **`CYLINDER`**
   - Given radius r and height h, calculate surface area A
   - Given volume V and radius r, calculate height h

Technical Requirements

- Server should be capable of accepting requests from **both** UDP and TCP clients.
- Server should support **multi-threading** (more than one client should be capable of using the cloud-calculator).
- Your protocol interaction should adhere to the following specifications.
- **Client Commands:**
  1. `HELO <server-hostname>` – This is the **first** command issued by the `client` ($\rightarrow$ `server`). The correct reply code (see section on reply codes below) `200`.
  2. `HELP` – This command can be issued anytime after the `HELO` command. The correct server reply code is `200`.
  3. `CIRCLE` – This command must be issued before any of the calculations related to the **`CIRCLE`**. The correct server reply code is `210`.
     (a) `AREA <r>` – The `AREA` command requests πr^2 calculation. The correct server reply code is `250`.
     (b) `CIRC <A>` – The `CIRC` command requests $2\pi\sqrt{\frac{A}{\pi}}$ calculation. The correct server reply code is `250`.
  4. `SPHERE` – This command must be issued before any of the calculations related to the **`SPHERE`**. The correct server reply code is `220`.
     (a) `VOL <r>` – The `VOL` command requests $\frac{4}{3}\pi r^3$ calculation. The correct server reply code is `250`.
     (b) `RAD <A>` – The `RAD` command requests $\frac{1}{2}\sqrt{\frac{A}{\pi}}$ calculation. The correct server reply code is `250`.
  5. `CYLINDER` – This command must be issued before any of the calculations related to the **`CYLINDER`**. The correct server reply code is `230`.
     (a) `AREA <r><h>` – The `AREA` command requests $2\pi rh + 2\pi r^2$ calculation. The correct server reply code is `250`.
     (b) `HGT <V><r>` – The `HGT` command requests $\frac{V}{\pi r^2}$ calculation. The correct server reply code is `250`.
  6. `BYE <server-hostname>` – This command closes the connection and requests a graceful exit. This command can be issued anytime during the interaction. The correct server reply code is `200`.
- **Server Reply Codes:**
  1. `200/210/220/230` Command Success. The command success reply code is issued only when the interaction happens according to the correct specification. Examples:
     – `200 HELO 10.1.2.3(TCP)` – If the `HELO` command is issued as the first command.
     – `200 <menu>` – If the `HELP` command is issued after `HELO`. the calculator menu is sent with this reply code.
     – `220 SPHERE ready!` – If the `SPHERE` command is issued at the correct point of interaction.
  2. `250 <answer>` – This reply code is issued in response to a correct calculator command syntax received in the previous message from client. `answer` is the calculated value.
  3. `500` – Syntax Error, command unrecognized.
  4. `501` – Syntax error in parameters or arguments.
  5. `503` – Bad sequence of commands.

Functional Requirements

1. IP addresses/hostnames and port numbers should not be hard coded.
   - Your server executable will accept a two command line argument as follows:
     `./server <tcp-port-number> <udp-port-number>`
   - Your client executable will accept two command line arguments as follows (assume your client to know the correct hostname port-number combo):
     `./client <server-hostname> <server-port>`
2. `client-server` connection can be either TCP (reliable) or UDP (unreliable). Your server should be able to accept both types of connections.
3. I will test with **at least** 2 simultaneous client connections, thus, your server should be multi-threaded.
4. Client's should exit gracefully. Server process is permitted to be forcefully killed.
5. Here's a sample (non-comprehensive) interaction. Assume the client's IP address is 10.1.1.2 and running UDP and the server's hostname is `calco`. Note: The server must recognize client IP and connection type.

Answer

Client		Server
HELO calco	→	
	←	200 HELO 10.1.1.2(UDP)
HGH 10 2	→	
	←	503 Wrong Command Order: CYLINDER before HGH
HELP	→	
	←	200 <menu-sent-back>
CYLINDER	→	
	←	230 CYLINDER ready!
HGH 10 2	→	
	←	200 0.8
BYE calco	→	
	←	200 BYE 10.1.1.2(UDP)

6. Your client and server should be able to run on two separate end systems. Bare minimum, you should verify an interaction between a client running on a lab machine (EB 1036 dual boots to Linux) and the "`cs home`" server and vice-versa. Depending on the firewall rules, you might also be able to test from off-campus using your own laptop/desktop as one end system as well.
7. At the end of your implementation, you should be able to:
   - Compile and run your code in a linux machine. Include a `README` file with clear compilation instructions and any non-standard Linux software requirements.
   - Run your server program first.
   - Run one or more clients to connect to the server.
   - Perform calculator functionality while meeting the technical requirements mentioned above.
   - Exit the client(s) gracefully.

Instructions

- **Start early!!**. There is quite a bit of functionality that needs to be implemented.
- **Take backups of your code often!!**. Use a good version control mechanism, if you don't have one already.
- You are required to use the socket API provided by your programming language. Use of pre-built network communication packages for these assignments will result in zero points. If you are unsure of what this means, talk to the instructor immediately (within the first week).
- Follow a good coding standard. Use one of Google Style Guides found here `https://google.github.io/styleguide/`, if you don't already follow one.
- The due date of this assignment is **Wednesday, February 14, 2018 @ 11:59:59 a.m.**. A dropbox will be opened for submission on Moodle.

Deliverables

A complete solution comprises of:

- A short report (max 5 pages) in **`.pdf`** of the design and implementation of your system. Your report should include the followings:
  - Introduction
  - Your own objectives of this assignment
  - Your design choices and/or specific implementation decisions.
  - The output of a sample run/walk-through of your solution (screenshots are highly recommended wherever applicable).
  - Summary and Issues encountered (if applicable).
- A short `README` file with compilation instructions and additional software requirements.
- A `makefile` to compile your code. Note: I will not attempt, nor entertain, any IDE-based compilations.
- A compressed tarball of the directory containing your source code directory, README, and makefile. **Do not** include executables in this tarball; doing so may result in penalties. To create a compressed tarball of the directory source, use the following command: `tar -zcvf name-pr1.tar.gz source/`.
  e.g. `tar -zcvf tgamage-pr1.tar.gz PR01/`.

Collaborating on ideas or answering questions is always encouraged. Most times, I find that you learn a lot from your peers. However, do not share/copy/duplicate code from others. If you inspire your solution based on code found online, remember to site their source in your report. Irrespective, do not submit someone else's code as yours. Issues related to academic integrity and plagiarism have **ZERO** tolerance.
Your Moodle dropbox will only accept **`.pdf`** and **`.tar.gz`**. There will be penalties for not producing deliverables in their acceptable file format.

Useful Resources

- Linux Man pages – found in all linux distributions
- Beej's Guide to Network Programming – A pretty thorough online tutorial found at `http://beej.us/guide/bgnet/output/print/bgnet_USLetter.pdf`