

CS447 : Networks and Data Communications

Programming Assignment #01

Total Points: 150

Assigned Date : Thursday, May 20, 2021
Due Date : Tuesday, June 01, 2021 @ 10:59:59 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:

- to get your feet wet on socket programming basics;
- to understand the ordering of the socket interface primitives;
- to get you exposed to Linux system calls (if you already haven't);
- to gain a basic understanding of network protocols; and
- 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 has a **rapid response** but not necessarily reliable; Haddock is fine with that as he is *all about speed*. 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:

- CIRCLE**
 - Given radius r , calculate area A
 - Given area A , calculate circumference C
- SPHERE**
 - Given radius r , calculate volume V
 - Given surface area A , calculate radius r
- 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 UDP 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 192.168.0.11(UDP) – 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. They are provided at runtime through configuration files[†] (e.g. `server.conf`) present in the same working directory.
 - Your server executable will accept a single runtime argument through a configuration file as follows:

```
./server server.conf
```

```
server.conf
UDP_PORT=
```
 - Similarly, your client executable will take two runtime arguments through a configuration file as follows:

```
./client client.conf
```

[†]Note: A configuration file is simply a text file with a `.conf` extension.


client.conf

```
SERVER_IP=
SERVER_PORT=
```

This are considered the default execution behavior for the servers and clients. Any deviations should get approved by the instructor first.

2. client-server connection should be UDP (unreliable) based.
3. We 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 192.168.0.11 and running UDP and the server's hostname is calco. Note: The server must recognize client IP and connection type.

Client	Server
HELO calco →	← 200 HELO 192.168.0.11(UDP)
HGT 10 2 →	← 503 Wrong Command Order: CYLINDER before HGT
HELP →	← 200 <menu-sent-back>
CYLINDER →	← 230 CYLINDER ready!
HGT 10 2 →	← 200 0.8
BYE calco →	← 200 BYE 192.168.0.11(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 one zone server container while the server is running on another.
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!!**. This is a fairly loaded assignment.
- **Take backups of your code often!!**.
- Follow a good coding standard. Use one of Google's coding standard found here <https://google.github.io/styleguide/>, if you don't already follow one.
- Your code must compile and run on a typical Linux setup. Neither the instructor nor his grader will use (or entertain the use of) any IDE to test your implementation. Be sure to test command line compilation and run before submission.
- **Absolutely do not** include executables, folders created by your programs, hidden files, version control repositories, or any other testing related files in this tarball. All project relevant file formatting standards (**PDF**, **README**, **.email**, **.txt**, **.tar.gz**) will be strictly monitored and is subject to penalties.
- The due date of this assignment is **Tuesday, June 01, 2021 @ 10:59:59 a.m.**. A Moodle dropbox will be opened for submission.
- Based on past student experience, multi-threading, especially handling multiple parallel UDP clients at the

server end, is not trivial. I suggest you to first get a single-threaded version working correctly and then think about extending it to multi-threading.

- This assignment can be fully developed using the socket API of your programming language and basic I/O API. Use of other packages/libraries without the instructors permission is not permitted.

Deliverables

A complete solution comprises of:

- A short report of the design and implementation of your system. The report should be **PDF** format. At a minimum, your report should include the following sections:
 - Introduction: Your objective and what you hope to gain from the assignment.
 - Overall design, specific design choices, and reply codes used.
 - The output of a sample run. Include plenty screenshots wherever applicable. In situations where we can't verify expected behavior, your screenshots maybe considered for partial credit.
 - Summary and Issues encountered. What you were able to achieve from your own objectives (from the introduction) as well as project specifications. Make sure to explicitly list functionality you failed to implement (or buggy).
- A compressed tarball that contains:
 - a directory containing (only) your source code and config files. **Do not** include executables, folders created by your programs, or any other files not specifically listed here as required.
 - A short README file with compilation and run instructions.
 - A makefile (**mandatory**) to compile your code especially if it involves compiling multiple executables with flag options. Python based submissions should use the makefile to echo the content of your README file.

To create a compressed tarball of the directory `source`, use the following command:
`tar -zcvf siue-id-pr1.tar.gz source/`
e.g. `tar -zcvf tgame-pr1.tar.gz PR01/`

Collaborating on ideas or answering each other's 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, online sources, or self-plagiarism from previous attempts. The exercise is meant for you to learn network programming, not to test your googling abilities. Issues related to academic integrity and plagiarism have **ZERO** tolerance.

Useful Resources

- Linux Man pages – found in all linux distributions
- Beej's Guide to Network Programming – A pretty thorough online tutorial found at https://beej.us/guide/bgnet/pdf/bgnet_us1_c_1.pdf