

# CS 447: Networks and Data Communications

## Project #01

Assigned Date : Tuesday, June 09, 2015

Due Date : **Tuesday, June 23, 2015 @ 10:14:59 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 yourself familiarized working with the socket programming basics;
- to understand the ordering of the socket interface primitives;
- to get your 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

Professor Calculus, returning from his latest conference, has just learned about the “power of the cloud” and now wants to move his scientific calculator application to “the cloud”. He has learned enough networking (to get by at least) and wants his application to support both **reliable** and **unreliable** connections. He also wants his cloud-based calculator to accommodate requests from **more than one person** at a time. Given that this is his first time network programming, he just wants provide support for only three basic operations:

- The power function (**POWER** ( $x^e$ )) for a given base  $x$  and an exponent  $e$ ;
- The cubic square root function (**CUBE** ( $\sqrt[3]{x}$ )) for a argument  $x$ ; and
- The factorial function (**FACT** ( $x!$ )).

## 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:**
  - 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. CALC – This command must be issued before any of the calculator functions (POWER/CUBE/FACT) can be used. The correct server reply code is 200.
4. POWER  $\langle x \rangle \langle e \rangle$  – The POWER command requests  $x^e$  calculation. The correct server reply code is 250.
5. CUBE  $\langle x \rangle$  – The CUBE command requests the cubic square root of a the given argument  $x$ . The correct server reply code is 250.
6. FACT  $\langle x \rangle$  – The FACT command requests the factorial value of  $x$ . The correct server reply code is 250.
7. BYE  $\langle server-hostname \rangle$  – 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 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 BYE 10.1.2.3(TCP) – If the BYE command is issued.
  - 200  $\langle menu \rangle$  – If the HELP command is issued after HELO. the calculator menu is sent with this reply code.
  - 200 CALC ready! – If the CALC command is issued at the correct point of interaction.
2. 250  $\langle answer \rangle$  – This reply code is issued in response to a correct calculator command syntax received in the previous message from client.  $\langle answer \rangle$  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  $\langle tcp-port-number \rangle \langle tcp-port-number \rangle$
  - Your client executable will accept two command line arguments as follows (assume your client to know the correct hostname port-number combo):  
./client  $\langle server-hostname \rangle \langle server-port \rangle$
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.

Client	Server
HELO calco →	← 200 HELO 10.1.1.2(UDP)
CUBE 64 →	← 503 CALC before CUBE
HELP →	← 200 <menu-sent-back>
CALC →	← 200 CALC ready!
CUBE 64 →	← 200 4
BYE calco →	← 200 BYE 10.1.1.2(UDP)



6. Your home folder is automatically mounted and accessible from all your experiment nodes. I advise you to create separate folders for each project to keep things clear of clutter.
7. At the end of your implementation, you should be able to:
  - Compile and run your code in Deterlab. Include a readme file with clear compilation instructions.
  - 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!!**
- **Take backups of your code often!!**
- Follow a good coding standard. Use the Google C++ coding standard found here <http://goo.gl/1rC1o>, if you don't already follow one.
- The due date of this assignment is **Tuesday, June 23, 2015 @ 10:14:59 a.m.**. A dropbox will be opened for submission on Moodle.

## Deliverables

A complete solution comprises of:

- A short report (max 5 pages) of the design and implementation of your system. Your report should include the followings:
  - Introduction
  - How to compile and run your software
  - The output of a sample run (including screenshots where applicable).
  - Summary and Issues encountered (if applicable).

- A short readme file with compilation instructions. Also preferable is a makefile to compile your code.
- A compressed tarball of the directory containing your source code. **Do not** include executables in this tarball. To create a compressed tarball of the directory `source`, use the following command: `tar -zcvf name-pr1.tar.gz source/`.  
e.g. `tar -zcvf tgame-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 use code found online, remember to site their source in your report. 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 [http://beej.us/guide/bgnet/output/print/bgnet\\_USLetter.pdf](http://beej.us/guide/bgnet/output/print/bgnet_USLetter.pdf)