CS 447: Networks and Data Communications

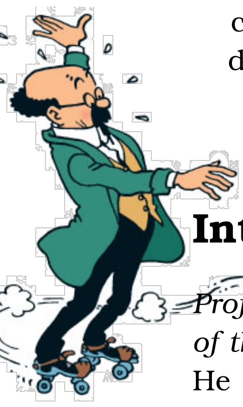
Project #01

Assigned Date	:	Friday, September 05, 2014
Due Date	:	**Wednesday, September 17, 2014 @ 02:59:59 p.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 yourself familiarized working with the socket programming basics;
b. to understand the ordering of the socket interface primitives;
c. to get your exposed to linux system calls (if you already haven't); and
d. to set yourself up for the rest of the course.



Introduction

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 be **reliable** and also for his cloud-based calculator to do accommodate requests from **more than one person** at a time. Since this is his first time networking programming he just wants provide support for only three basic operations:

1. Compute square root (***sqrt***) for a given number;
2. Compute the base-e exponential function(***expn***) of a given number; and
3. Compute the factorial (***fact***) of a given number.

Tintin meanwhile learns of Prof. Calculus's intents and (as usual) is willing to help. He let's prof. Calculus borrow one of his old partially completed socket programs from his networking class, hoping he can modify it for his purposes.

Instructions

- **Start early!!**
- **Take backups of your code often!!**.

- Grab the stub code found at `www.cs.siue.edu/~tgamage/CS447/assigns/gamage-111-pr1.tar.gz`
- Fill in the missing system calls. Get it to run. This is an echo server where the server simply echoes what the client sends. Hint: use the linux programmer's Manual pages to find and learn the corresponding systems calls; e.g. to find information about the `socket` system call, simply type `man socket` on your linux terminal.
- The stub code, once completed runs as follows:
  - To compile `server.c`

    ```
    $ gcc server.cxx -o server
    ```
  - To run `server`

    ```
    $ ./server
    ```
  - To compile `client.c`

    ```
    $ gcc client.cxx -o client
    ```
  - To run `client`

    ```
    $ ./client <server-name>
    ```
  - Technically, both `server` and `client` can be run on a single machine. In such cases, especially for your testing, your <server-name> argument would be `localhost`

- Modify the stub code `client.c` as follows:
  - The client runs as

    ```
    $ ./client <server-name> <server-port>
    ```
  - `<server-name>` is the hostname of the server (e.g. localhost) and `<server-port>` is the server's port number to connect.
  - Once connection is established the following menu is presented at the client's interface

    ```
    $ (S) - Square root <arg>
      (E) - Exponential <arg>
      (F) - Factorial <arg>
      (X) - Exit
    ```

    where the client will make a selection with the corresponding argument; e.g. to find a the square root of `16`, the command would be `S 16`
- Modify the stub code `server.c` as follows:
  - The server run as

    ```
    $ ./server <port>
    ```
  - Here, `<port>` is the port number which the server is listening on.
- The `server` once a request from the `client` is received, responds promptly with the corresponding answer (or an error message), which gets printed on the `client's` standard output

- You are to assume that the client knows the IP address of the server. Technically, you can develop your code to run on a single machine (through localhost), but hostnames and port numbers should not be hard coded to your solution
- You may use any programming language of your choice, if you don't want to do this assignment in C. However, you should make sure that your code compiles and runs on any typical Linux machine. Hint: EB 1036 (first floor computer lab) has dual-boot capable lab computers that you may wish to use.
- Follow a good coding standard. Use the Google C++ coding standard found here `http://goo.gl/1rC1o`, if you don't already follow one.

Deliverables

The due date of this assignment is **Wednesday, September 17, 2014 @ 02:59:59 p.m.**. A dropbox will be opened for submission on Moodle. 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 run your software (Remember! I give you the liberty to choose your programming language of choice. Don't expect my system to have all the software needed to run your program. I do currently support C, C++, Java, and Python on my Linux-based machine. If your choice of programming language is not one of these, make sure to mention what and how I need to install whatever that is needed to compile and run your program)
  - The output of a sample run (including screenshots where applicable)
  - Summary and Issues encountered (if applicable)
- 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-111-pr1.tar.gz source/`. Obviously, change the name to your last name and 111 to the last three digits of your SIUE ID.

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`