CS351 Lab #4

Network I/O

Instructions:

- *Assigned date: Thursday November 4th, 2020*
- *Due date: 11:59PM on Friday November 13th, 2020*
- *Maximum Points: 40*
- *This lab must be done individually*
- *Please post your questions to the Piazza forum*
- *Only a softcopy submission is required; it will automatically be collected through GIT at the deadline; email confirmation will be sent to your HAWK email address; submissions will be graded based on the collected submissions at the deadline, unless an email to the TAs at cs351-ta-group@iit.edu with the subject "[CS351] homework submission is delayed" is received; when a student is ready to have their late assignment graded, they must send another email to the TAs at cs351-ta-group@iit.edu with the subject "[CS351] late homework submission is ready"; late submission will be penalized at 5% per day*

1 Your Assignment

This project aims to teach you about network I/O. You must use the C programming languages. You can use the following header files; no other libraries or header files can be used to complete the assignment, without express permission.

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
#include <wait.h>
#include <time.h>
#include <sys/types.h>
#include <rpc/rpc.h>
```

Libraries such as STL or Boost cannot be used. You can use any Linux system for your development, but you must make sure it compiles and runs correctly on fourier. The performance evaluation should be done on your own computer in a Linux environment (e.g. think back at Lab #0). Given the large class size, we encourage all students to avoid using fourier for anything but testing functionality on small datasets.

In this project, you will evaluate the cost of invoking a set of simple functions between two processes, and compare it to the performance of a local function invocation. You will explore local function calls, pipe, sockets, and RPC in this assignment. A scheleton codebase has been implemented that implement the addition, subtraction, multiplication, and division of two double values. You can compile the code with make, and test the code with the following command:

```
./netio function add 1000000
```

```
* running netio with method function operation add for 1000000 number of ops...
==> 108003024.084674 ops/sec
```

Usage information can be found below:

```
./netio
usage: ./netio <method> <operation> <num_ops>
    - method: function / pipe / socket / rpc
    - operation: add / subtract / multiply / divide
    - num_calls: 1000 | 1000000
```

Timing code has already been included to measure the time elapsed, and it computes the number of operations per second achieved.

You must implement these four functions (add, subtract, multiply and divide) as remote methods. You are to communicate between the client process (that invokes these functions) and the server process (that receives the function information, and executes the function call, and returns the value) through 3 different mechanisms: pipes (either named or unnamed are fine), sockets (TCP/IP), and remote procedure calls (RPCGEN). You can assume the two processes will be on the same computer. You can also assume that there will be only 2 processes, 1 client and 1 server that must be supported at any one time. Keep the client and server as simple as possible. Fill in the table below with performance data.

method	**num_ops**	**+ ops/sec**	**- ops/sec**	*** ops/sec**	**/ ops/sec**
Function	1 billion				
Pipe	1 million				
Socket	1 million				
RPC	1 million				

Other requirements:

- You must write all benchmarks from scratch. Do not use code you find online, as you will get 0 credit for this assignment.
- All benchmarks must be compiled and tested for functionality on fourier.
- All benchmarks can be run on your own Linux system or on fourier (both are fine)
- Since there are many experiments to run, you are encouraged to use a bash script to automate the performance evaluation; however, this bash script is not mandatory as it was in prior assignments
- No GUIs are required. Simple command line interfaces are required.

3 What you will submit

When you have finished implementing the complete assignment as described above, you should submit your solution to your private git repository. Each program must work correctly and be detailed in-line documented. You should hand in:

1. **Source code and compilation (80%):** All of the source code in C and Bash; in order to get full credit for the source code, your code must have in-line documents, must compile (with a Makefile), and must be able to run a variety of benchmarks through command line arguments. Must have working code that compiles and runs on fourier.
2. **Report / Performance (20%):** A separate (typed) design document (named lab4-report.pdf) describing the results in a table format. You must evaluate the performance of the various parameters outlined and fill in the table specified to showcase the results. You must summarize your findings and explain why you achieve the performance you achieve, and how the results compare between the various approaches. Highlight the best one in green, and worst one in red. Can you explain in words why they are the best, or the worst, and why the data you have makes sense.

To submit your work, simply commit all your changes to the Makefile, netio.c, and any other files (e.g. server code for pipes, sockets, and additional RPCGEN files) you created to support this lab, and push your work to Github. You can find a git cheat sheet here: https://www.git-tower.com/blog/git-cheat-sheet/. Your solution will be collected automatically at the deadline. If you want to submit your homework later, you will have to push your final version to your GIT repository and you will have let the TA know of it through email. There is no need to submit anything on BB for this assignment. If you cannot access your repository contact the TAs.

Grades for late programs will be lowered 5% per day late.