EECS 338 Homework 5: Interprocess Communication

General requirements:

- Due on the posted due date.
- Upload a single, compressed file (e.g. zip) to Canvas that contains all required files.
- Include either a single makefile that compile all programs or a separate makefile for each.
- Include a typed document with a description of any techniques that you used to control the output order.
- All work should be your own, as explained in the Academic Integrity policy from the syllabus.

Instructions: The purpose of this assignment is to work with sockets, pipes, and threads.

1. Modify server.c and client.c such that the server sends the client a question, receives the client's answer, and sends the client the correct answer with a message indicating whether the client was correct. The server should print one message to indicate when it is listening and a second message to indicate whether the client's answer was correct. The client should print the server's question, allow the user to type an answer, and print the server's response. If desired, you may use a fixed port number and IP address of 127.0.0.1. Include output in your report for two cases: where the client answers correctly, and where the client answers incorrectly. Below are two examples.

   Example where client answers correctly...

Server:	Client output (user input in **bold**):

`$ ./server`
`Listening for client...`
`Client was correct.`
	`$ ./client`
	`How many steps does it take for a client to change a light bulb?`
	`3`
	`Correct! The 3 steps are: create a socket, connect to it, and send a new bulb.`

   Example where client answers incorrectly...

Server:	Client output (user input in **bold**):

`$ ./server`
`Listening for client...`
`Client was NOT correct.`
	`$ ./client`
	`How many steps does it take for a client to change a light bulb?`
	`1`
	`Nope. The 3 steps are: create a socket, connect to it, and send a new bulb.`

2. The program "shubert.c" demonstrates the use of the Shubert function, which is a test case for optimization algorithms. Using fork() and pipe communication, find the minimum value of the Shubert function for $-2.0 \leq x_1, x_2 \leq 2.0$. Use two processes such that each performs an equal amount of work to find a local minimum in its range. For example, one process can do half of the x_1 values, and the other process can to the other half. The parent process can determine which of the two local minima is the least and print it as the global minimum. Include the output in your report.
3. Repeat #2 using POSIX threads with one child (two threads total). Include the output in your report. Additionally, explain in 2 - 3 sentences whether you like #2 or #3 as a solution to this analysis problem.

Interested students can find more information and graphs of the Shubert function here:

- http://www.sfu.ca/~ssurjano/shubert.html
- http://profesores.elo.utfsm.cl/~tarredondo/info/soft-comp/functions/node28.html

Tips:

- *Compiling on eecslinab servers:* To use the "cos" function on an eecslinab server, you need to link the math library using "-lm". For example: `gcc -o prog prog.c –lm`
- *What output is required for Shubert problam?* Only the global minimum value is required. The demonstration program shubert.c prints the individual values, but that is not desired for the analysis. You may wish to print the values from the parent and child just to see what they are, but this is not required. As a realistic application, you may also wish to try very small intervals (much smaller than 0.5), which is where parallelization becomes useful by dividing the work.
- *Global variables:* For threads, you can use global variables (you do not shared memory as in the previous assignment).
- *Strings vs. doubles in pipes:* One example in class for pipe communication (unix_pipe.c) used strings for message passing. The example program pipe_double.c provided in class demonstrates how to use doubles.

Rubric:

Item	**Points**
Makefile(s)	5
1: Socket setup for server	10
1: Socket setup for client	10
1: General server writing/reading (not including sync)	10
1: General client writing/reading (not including sync)	10
1: Synchonization of server/client (proper read/write)	10
2: output	5
2: creating child process	5
2: dividing up search for minimum	5
2: pipe communication	5
3: output	5
3: creating child thread	5
3: dividing up search for minimum	5
3: thread communication	5
3: Discussion for explanation of which is preferred (2 or 3). Any reasonable explanation will receive full credit.	5
Total	100