**Laboratory 4  
Due: 13 Oct 2016 (through Blackboard)**

This laboratory will result in your ability to build a simple program that passes data structures in messages between a client and server program.

**Objective**:

Write a client that sends a message containing two integers and a character representing the operation to a server. The server computes the numeric value for the result of the numbers and the given operation, and then returns a message containing a structure as the result. The client will pass the two numbers as integer members of an “input parameters” struct and the operation as a character value in the struct. The server will reply with a different struct containing a double, a flag representing whether or not an error occurred (or would have occurred if the operation had been attempted), and a character string of up to 128 bytes containing the error description. (You can use your imagination for the string content, but try to cover all types of errors.)

The operation should be one of ‘+’, ‘-‘, ‘x’, and ‘/’. The input integer values could be any valid integer. The server must check for possible overflow, and for division by 0.

You should start the server in the background (using &) and have it print out its process\_id. Then multiple client applications could be run in the foreground that use a command-line argument on the client application to pass the PID of the server which they want to use to perform the operation.

The client program will block until receiving the result message from the server program and then the client program will output the result (or error condition) to the console. You can find an example below.

**Example:**

#**./lab4\_server &**

The server is running as process\_id 77234.

#**./lab4\_client 77234 6 + 9**

The server has calculated the result of 6 + 9 as 15.

## Steps:

1. Create one or more new projects named “Lab4\_xxxx” inside the GIT repository for your laboratory work.
2. Create an include file “mystruct.h” to define the typedefs for structs to be used for sending requests and receiving responses.
3. Create the *lab4\_server* and *lab4\_client* programs.
4. Test your programs.
5. Save and commit the changes to your repository.
6. Go to the working directory for your GIT repository (the one named with your Algonquin username) and zip it.
   1. If you open the “Window>Open Perspective>Other…>Git Repository Exploring”, you can see where it is physically stored.
7. Submit your laboratory for grading by demoing it to the professor in the laboratory and then submitting it online using Blackboard under the “Handing in Work” link, with the zip file as an attachment.

## Grading Scheme

Marked out of 30.

|  |  |
| --- | --- |
| **Criteria** | **Mark** |
| Superior capability. Submission meets or exceeds expected standards | 30 |
| Satisfactory capability, acceptable product/result | 20 |
| Marginal capability, substandard product/result | 10 |
| No capability, unacceptable product/result. Work not submitted | 0 |