Name: Sean Bartholomew
Date: 20160702
Current Module: Network Programming with C
Project Name: FDR

Project Goals:

The goal of this project is to create a UDP server listening on multiple ports, that is capable of parsing input and running multiple functions based on the input.

Considerations:

What language should the program be written in
How will the program support decimal numbers up to 10^30
What is a valid roman numeral
How should invalid data be handled

Initial Design:

The requirement to support decimal numbers up to 10^30 solidified my decision to use python (C does not easily support numbers that high). The requirement for multiple ports means that the program will have to have multiple threads. The input as a string means that regular expressions will have to be used to verify the input is in a valid form. The entire input MUST be valid, if there are unexpected characters the data will be ignored. If the roman numeral is not in the correct order it is assumed to be invalid data. There will need to be three functions to support the three required mathematical operations. In order to stop the threads once the server is stopped the threads cannot be blocking on a read operation.

Data Flow:

When the server starts it identifies the UID, and creates three threads to listen on ports UID, UID + 1000, and UID + 2000. Each server listens on its respective port. When data is received, it is evaluated using regular expressions if the input is valid the appropriate function is performed on it. If not a message is sent back to the client saying bad input. If the data was valid, the result is returned to the client in hexadecimal.

Communication Protocol:

CMD line options: none. The ports are already specified.
Signals handled: None.
Potential Pitfalls: I could have handled signals better.

Test Plan:

Make sure the server works. The real difficulty in the project is ensuring that the program will work every time. Because threads do not schedule things the same way every time, errors are often difficult to identify and more so to fix.

Expected Result:

I expect the behavior outlined in the initial design to be appropriately implemented. The program will not crash.

Conclusion:

The server is not nearly as difficult as I thought it would be. I probably should have commented more in the code. Also, 900 in roman numerals is CM not DCD – the latter is not a valid combination. For reference on how the program handles Roman numerals: https://www.math.nmsu.edu/~pmorandi/math111f01/RomanNumerals.html