Assignment 1

CPSC 416

Due January 21

The purpose of this assignment is to create a multi-threaded server. The server needs to be written in C using Pthreads. The server should be called "mtserver" and it should take two parameters (a) the maximum number of clients (N) that can connect, (b) the server port number. You need to provide a "Makefile" that creates an executable called "mtserver". Your makefile (with make not only gmake) needs to be able to compile your program on a standard **Linux** department server. We will be automating the compilation and execution of your server so you MUST follow the instructions. We will use the "handin" command for submitting this assignment. The assignment is midnight (11:59PM) on the date given.

The main server thread does the following:

1. Create a listening socket
2. Perpetually look, doing the following:
   a. Accept a new connection from a client
   b. If the current number of connections is greater than N, immediately terminate the connection, else start a new child thread to respond to queries.

A child thread should do the following:

1. Read data from the new connection.
2. Process the request. There are three possible requests:
   a. "uptime", return the integer value of the Unix clock.
   b. "load", returns the integer value of the current number of clients.
   c. "exit", returns value 0 and goes to Step 4 below. Notice that all returned values are 4 byte integers.
   d. Anything other than these three requests is considered invalid and -1 is returned. You should terminate the client process when a client makes more than two consecutive invalid requests.
3. Return to 1, looking for another request.
4. Close the connection to the client and terminate the child process.

Note the protocol assumes lower case ASCII character commands uptime, load and exit and returns a 4 byte integer. The commands are not "\0" terminated.

I recommend first creating a simple single threaded server. You will also need a client to interact with your server. Once you are satisfied that you have the basics working then you can extend it with Pthreads to start up a child process to respond to queries. At this point to keep it simple, simply return a bogus value for the load and also do not worry about keeping track of the number

of clients. Please make sure to get at least this far on the assignment! You need to get this far to obtain a passing grade on the assignment.

Now you are ready to tackle the more challenging parts of the assignment.

1. You need to declare a shared variable to keep track of the number of connected clients. You will need to guarantee "mutual exclusive" access to this variable using Pthread function calls.
2. You can work on the reliability part of the assignment to ensure clients cannot crash servers. This is somewhat open-ended and we will try to provide you with clients you can run against your server. A couple of simple cases that come to mind are buffer overruns and unexpected client disconnects.

Often it is not necessary to re-invent the wheel, it is good to start from an existing working program. As a help, you are free to use any code by Donahoo and Calvert from their book TCP/IP Sockets in C (http://cs.baylor.edu/~donahoo/practical/CSockets) , which is a great introduction to Sockets for beginners. You can also use Beej's Guide to Network Programming (http://beej.us/guide/bgnet/output/print/bgnet_A4.pdf) and code from there. Make sure that you understand any of the code you use (just blindly copying code isn't very helpful, even typing it in yourself allows you to at least know what needs to be included). Also make sure in the header you acknowledge the source of any code you have used! DO NOT use any other code sources!

The bible for network programming for sockets is "UNIX Network Programming : The Socket Networking API", Volume 1 (THIRD EDITION), by Richard Stevens, Bill Fenner, and Andrew Rudoff. You will also want to look at tutorials for Pthreads.