Multi-threaded Reverse Echo Server

In this project you will implement a multi-threaded server that reverses a string provided by a client. The objective of this project is to provide you with hands on experience with socket programming and multi-threading. Your server will wait on a given port (specified as command line parameter). Client will open a connection to the server and send a string. The server should create a new thread to handle each incoming connection. The thread will read the string, reverse it and send back the reversed string.

Specifically, your server should do the following:

1. Wait on a specified port (the port number is specified as a command line parameter).
2. Upon client connection, the server should create a new thread and hand over the connection to the new thread.
3. The main thread should continue to wait for other incoming connections.
4. The connection handling thread should read the string sent in by the client.
5. The connection handling thread reverses the string.
6. The connection handling thread logs the following information onto a log file – IP address of the client requesting service, the thread ID handling the request, the string sent by the client and the response sent back to client.
7. The response is sent back to the client
8. The connection is closed.

Points to be noted:

1. You should use TCP sockets.
2. You can write your server in Java/C++/C. If using Java, you should use socket class. If using C++/C, you should use the PThreads package.
3. Your program has to be multi-threaded. If not, there will be an automatic deduction of 50 points (i.e., the maximum points you can get will be 50).
4. Irrespective of which language you use, your server should work with our client (implemented in Java). You can turn in your test client if you want to. It may help TA to evaluate the project. Tests will be performed on a Linux machine.
5. You can assume that the request from the client is terminated with a new line character (i.e., the client request is single line).
6. The server will continue to execute until it is explicitly terminated (for e.g., by killing).
7. The log file at the server end should be named revecholog.txt

What to submit:

A zip folder containing:

1. All source files (including any .h files) of server.
2. Optionally, all source files of client.
3. makefile
4. Readme file containing: (1) project partner information; (2) Any special program execution instructions

Submission to be done via ELC.