Testing:

For testing, mainly used curl to check the functionality of the program. The first thing I implemented was the multithreading functionality for the program. In order to test this, I ran multiple terminals with very large files and tried GET and PUT commands and see if they ran at the same time. After those were working, I used a bash script to run around 20 requests at the same time with less threads. After that was working, I moved onto logging where I tested nearly the same way. I first started with small tests on my logging function by making direct calls to the function with main. After I ensured the functionality, I moved the code into the rest of my program to be used with GET/PUT requests. I used curl to send requests and checked the log files to see if I got the desired outcomes. I also used hexdump -C in order to see if the hex values in my file were the correct ones. After getting logging to work, I did further testing using the bash script to see if the logging also works with multiple threads.

Question:

Using either your HTTP client or curl and your original HTTP server from Assignment 1, do the following:

- Place four different large files on the server. This can be done simply by copying the files to the server's directory, and then running the server. The files should be around 4 MiB long.
- Start httpserver.
- Start four separate instances of the client at the same time, one GETting each of the files and measure (using time(1)) how long it takes to get the files. Perhaps the best way to do this is to write a simple shell script (command file) that starts four copies of the client program in the background, by using & at the end.

Q. Repeat the same experiment after you implement multi-threading. Is there any difference in performance?

A. After repeating the same experiment after implementing multithreadings, there is not too much difference in performance. There are the occasional runs that are very fast but on average, the times are not all too different. They both range around 0.015s to 0.60s and fluctuate between both.

Q. What is likely to be the bottleneck in your system? How much concurrency is available in various parts, such as dispatch, worker, logging? Can you increase concurrency in any of these areas and, if so, how?

A. I think there are two likely bottlenecks in my system. One guaranteed bottleneck is the way my program does logging one character at a time. That introduces a ton of slow down into my system as it has to process every single character. The other bottleneck

might be cpu performance as the cpu works hard running multiple threads and processing this much data. The concurrency in my system is with the dispatch and the worker threads. They can all run at the same time however my logging does not run concurrently. I can increase concurrency in this area by having a seperate thread that is made just for logging. That way my worker threads can be working on processing the requests while another thread is logging data.