Alexander Soe
asoe@ucsc.edu
1533280

Asgn1 Writeup

Most of my testing involved unit testing with each portion of the code. The way I would test my code was through "curling" different responses to my server. I would curl for GET requests and PUT requests initially by having files on each side of the socket and checking each side for the specific file that was supposed to be put there respectively. I would then curl for things such as if a PUT request had no content length and also for error statuses which involved using improper ports and addresses, as well as improper resource names. The one thing that really troubled me was PUT requests with no content length. My code wouldn't stop reading as it could not obtain the EOF. The way I was able to debug this was through running it through the Xcode debugger. I was able to figure it out and fix it by sending a status message after I was able to get the first read, and then writing the contents of the buffer to the file that I opened. Lastly, to check for multiple GET and PUT requests I used netcat to debug. One of the problems I found doing multiple commands was returning the correct content length on a GET Command. It turned out that it was how netcat read the contents of the file that held all the commands.

The server will try to read chunks of the code up until the loss of connection. After each chunk is read, they will be written, and another chunk will try to be read until the buffer is full. When the connection is lost, whatever is in the buffer will be written and no status message can be written since read will return zero and the server socket will close. This is different than "dog", because all the files and content are not local. We don't have all the information from the client immediately and when the connection is lost, we are unable to obtain whatever information we were unable to obtain through read. The complexity is the fact that we aren’t reading from a local file but rather a data transfer over clients and a server. The data isn’t going anywhere in the “dog” assignment, whereas the data in this assignment can be lost to either the client or server during a transfer due to a connection loss during a read.