Assignment 1: HTTP Server **Writeup** **Emil Ingemansson (eingeman)**

1) What fraction of your design and code are there to handle errors properly? How much of your time was spent ensuring that the server behaves "reasonably" in the face of errors?

    a) Almost half of my code is there to handle errors from the user. First, I have a function dedicated to parsing through the address sent by the user to make sure that it follows the allowed upper and lower case, 0-9 and - _ ruleset. Furthermore, in both my get and put, there are multiple if statements leading to different scenarios where the server will return an error such as 404 where the file is not found. I'd say a good amount of time was spent making sure the errors were handled properly because in general this assignment would have been very similar to assignment 0 if we did not have to deal with the errors from the client, reading and writing to and from file descriptors.

2) List the "errors" in a request message that your server must handle. What response code are you returning for each error?

    a) 400 bad request - The server received a bad request. This most likely happens when an address that is not allowed it sent to the server. This error could also be sent back if the client sends an http header that doesn't end in \r\n\r\n.
    b) 403 forbidden - This error code would be sent back if the file that the server is trying to access for the client is not allowed. This can happen if another server is already using the file because both servers are not allowed to edit the file at the same time.
    c) 404 not found - This error is returned in get requests where the client sends a valid request but no data can be found at that address specified by the client.

3) What happens in your implementation if, during a PUT with a Content-Length, the connection is closed, ending the communication early?

    a) If during a PUT and the connection is closed prematurely (server is shut down), the server will not send back a message to the client (empty reply) but since the sockets are not closed, the connection will still be intact. A file will be created with the address specified by the client if it was valid but the data will not be loaded into it because the connection was ended early.

4) Does endianness matter for the HTTP protocol? Why or why not?

    a) Endianness does matter for the HTTP protocol because the client and server both need to agree with how they are communicating with each other. If the client sends its message in little endian and the server expects to be reading a message in big endian, nothing will work. The way the server and client communicate across the sockets must be in agreement.