Ricky Zhao

rjzhao@ucsc.edu

Testing Receiving the header:

Test1:

Server: ./httpserver localhost 8080

Client: curl -v localhost:8080/

This test is to test if the header is received correctly. I received the header to a buffer and printed out the buffer.

Testing GET function:

Test 1:

Server: ./httpserver localhost 8080

Client: curl -v localhost:8080/test

This scenario is to test the function of GET. This test is done before implementing the function that checks if the filename is valid. I checked if the content is correctly printed to standard output.

Test 2:

Server: ./httpserver localhost 8080

Client: curl -v localhost:8080/nofile

nofile is a non-existent in the server's directory. This is to check if my program catches this error and sends the proper error messages.

Test 3:

Server: ./httpserver localhost 8080

Client: curl -v localhost:8080/permission_test1234567890__

permission_test1234567890__ has it's read and write permissions turned off. This test is to check if the program will catch this error and send a code 403 Forbidden error back to the client.

Testing PUT function:

Test 1:

Server: ./httpserver localhost 8080

Client: curl -v -T put_test localhost:8080

This scenario is to test the function of PU. This test is done before implementing the function that checks if the filename is valid. I checked if the file is created in the server's directory and properly written.

Test 2:

Server: ./httpserver localhost 8080

Client: curl -v -T put_test localhost:8080

put_test have been modified on the client side. This test is to check if the program properly overwrites the file on the server side.

Test 3:

Server: ./httpserver localhost 8080

Client: curl -v -T permission_test1234567890__ localhost:8080

permission_test1234567890__ has it's read and write permissions turned off. This test is to check if the program will catch this error and send a code 403 Forbidden error back to the client.

Testing valid filename:

Test 1:

Server: ./httpserver localhost 8080

Client: curl -v  localhost:8080/ABCDEFabcdef012345XYZxyz-mm

This test is to check if the program recognized that this is a valid filename

Test 2:

Server: ./httpserver localhost 8080

Client: curl -v  localhost:8080/ABCDEFabcdef012345XYZxyz-m/

This test is to check if the program recognized that this is an invalid filename and sends the proper error messages.

Test 3:

Server: ./httpserver localhost 8080

Client: curl -v  localhost:8080/test

This test is to check if the program only recognized filenames of only 27 characters.

Testing socket creation

Test 1:

Server: ./httpserver

This test if the default port is port 80

Test 2:

Server: ./httpserver localhost 8080

This test if the checks if the program handles more than 1 arguments and sets it to the right port.

Test 3:

Server: ./httpserver asdf 8080

This test if the program catches invalid addresses.

Write up question:

If during a PUT the connection was closed early, no files would be written since my program only does one receive per PUT request. Since no data will be received, no data would be written to a file. In dog, we are not sending information through a socket so there is no risk of communication ending early. In dog you are passing the file as an input and working in the same directory. Therefore, there this extra concern is not presented in dog.