Julia Sales
ID: jesales
Asgn1
CMPS130-02

Write Up

Testing

For testing my http server. I would test it on my local machine, on my school ssh account, and on Ubuntu. I would keep testing small portions of my code while updating the design document as well as writing the code. I had to constantly go back and forth between my design document and code because I could not get it right the first time. I tested in a similar way for linux and for Ubuntu. I first started using curl to test one client at a time to make sure each small part of the programming assignment worked. I then used netcat to test multiple clients to my server on all my platforms.

Problems

There are a few problems with my code. PUT for large files does not work 100% of the time. Sometimes it will work, sometimes it will not work. I am not sure why.

Questions:

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?

Answer: A good two-thirds of my code handle errors. To make sure that the server behaves reasonably, I looked on the wiki page the professor gave us and learned about what each status code meant. I tried to apply as many test cases as possible for my program. A majority of my program should work properly for majority of errors.

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

Answer:

GET

- target names with "/": HTTP/1.1 403 Forbidden
- target file does not exist: HTTP/1.1 404 File Not Found
- target name is not 27 ascii characters: HTTP/1.1 400 Bad Request
- a file is "gotten": HTTP/1.1 200 OK
- If non of these apply in GET: HTTP/1.1 500 Internal Server Error

PUT

- target name with "/": HTTP1.1 403 Forbidden
- target name is not 27 ascii characters: HTTP/1.1 400 Bad Request
- if file does not exist: HTTP/1.1 201 Created

- if file does exist: HTTP/1.1 200 OK
- if non of these apply in PUT: HTTP/1.1 500 Internal Server Error

What happens in your implementation if, during a PUT with a Content-Length, the connection is closed, ending the communication early?
Answer: In my code, not all of the contents of the file gets put into the target code. It will say Recv failure.

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

Answer: Which ever endianness is used does not matter, however it has to be consistent

Whatever endianness is used prior to the transfer and after the transfer must be the same

endianness.