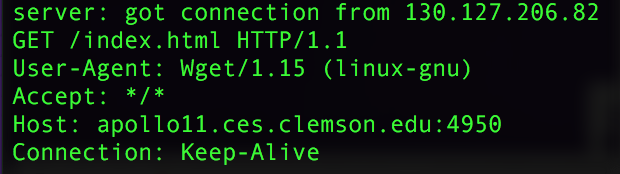
ECE 6380 Computer Communications: Machine Problem 1 Questions

1. Do the pairs of programs interfere with each other? Explain why or why not.  
   The two different sets of programs do not interfere with each other even though they are bound to the same port, as each process forks off and runs on its own once it has bound to the port so each process runs independently regardless of the original port that it binds to, as well as the fact that there is no connection being made between the listener and talker in UDP as there is in TCP, therefore there also should not be an issue in this case for that reason either.
2. Print a copy of the information the server receives. Identify the START\_LINE and the MESSAGE\_HEADER that your new server receives from the browser. See the description of the request and response messages in section 9.1.2, identify these fields in your print out.  
     
     
   The START\_LINE is the line beginning with “GET /index.html HTTP/1.1”  
   Each different line containing a category and then a colon is a different MESSAGE\_HEADER for a different piece of information.
3. This part should briefly describe the implementation history for your client and server programs for part c. In particular, you should briefly discuss (a) how the server protects itself from a client that sends invalid information; (b) what is the limit on the size of the largest string that the server can send; (c) how does your code ensure that the client has sufficient memory to receive the string;(d) how does the client protect itself if the server incorrectly sends a string that is too big. In addition, (e) you should acknowledge the sources if you copied sections of your code from outside sources.   
   1. In my implementation the server protects itself from a client that sends invalid information by entirely disregarding that client’s message after it has been deemed invalid and returning a message to the client informing it of the error.
   2. The limit on the size of the largest string the server can send is the technically 100 as there is a defined value for the largest number of bytes that can be sent or received at once, so that only 100 bytes can be received by the client before the buffer overflows and loses information that is sent.
   3. By the same logic, the code given ensures that up to 100 bytes are available for the client to receive a string.
   4. The client protects itself from the server sending a string that is too big by only allowing a maximum amount of bytes of 100 before it just throws away more information, so that extra information is not written to memory as to cause true buffer overflow problems.
   5. The outside sources that code was either edited from or used were from Brian “Beej” Hall, as well as tutorialspoint.com’s page on the strspn() c library function, and stackoverflow was used for other syntactical issues with the network system calls.