CS536 - Spring 2017
Programming Assignment 1
Web Server

Due Date: 2nd Feb, 2017 11:59 pm

I. Objectives

In this lab, you will learn the basics of socket programming for TCP connections in Python: how to create a socket, bind it to a specific address and port, as well as send and receive a HTTP packet. You will also learn some basics of HTTP header format

II. Getting Started

A. Socket Programming

https://docs.python.org/2/howto/sockets.html

B. HTTP Protocol

https://tools.ietf.org/html/rfc1945section-8.1

III. Implement

A. Programming environment

You must work individually on this assignment. You will write Python code that compiles and operates correctly on the XINU machines (xinu1.cs.purdue.edu, xinu2.cs.purdue.edu, etc.) found in HAAS 257.

B. Web Server

You will develop a web server that handles one HTTP request at a time. Your web server should accept and parse the HTTP request, get the requested file from the servers file system, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server, the server should send an ẗHTTP 404 Not Found¨ message back to the client. If a file is present but the proper permissions are not set, a permission denied error is returned.

You should also note that web servers typically translate ”GET /” to ”GET /index.html”. That is, index.html is assumed to be the filename if no explicit filename is present.

Your program should accept the port number as a command line argument. Please name your source file server.py. For this assignment, you will need to support enough of the HTTP protocol to allow an existing web browser (Firefox, Safari or Chrome) to connect to your web server and retrieve the contents of sample pages from your server. (Of course, this will require that you copy the appropriate files to your server’s document directory - please name the server document directory as Upload). We will be mainly checking for txt files and images.

At a high level, your web server will be structured somewhat like the following:
Forever loop :
Listen for connection

Accept new connection from incoming client
Parse HTTP request
Ensure well-formed request (return error otherwise)
Determine if target file exists and if permissions are set properly (return error otherwise)
Transmit contents of file to connect
Close the connection

Currently, the web server handles only one HTTP request at a time. Implement a multithreaded server that is capable of serving multiple requests simultaneously. Using threading, first create a main thread in which your modified server listens for clients at a fixed port. When it receives a TCP connection request from a client, it will set up the TCP connection through another port and services the client request in a separate thread. There will be a separate TCP connection in a separate thread for each request/response pair.

Instead of using a browser, write your own HTTP client to test your server. Your client will connect to the server using a TCP connection, send an HTTP request to the server, and display the server response as an output. You can assume that the HTTP request sent is a GET method. The client should take command line arguments specifying the server IP address or host name, the port at which the server is listening, and the path at which the requested object is stored at the server. The following is an input command format to run the client.

```
client.py server_host server_port filename
```

The client should save the files in the folder named as Download.

IV. Submission and Grading

A. What to Submit

You will be submitting your assignment on blackboard. Your submission directory should contain contain:

- All source files.

Note:

- Please document any reasonable assumptions you make or information in this file, e.g., if any parts of your assignment are incomplete.

*Questions about the assignment should be posted on **Piazza**.*