CSc361 Spring 2023 Programming Assignment (P1): Simple Web Server (SWS) Specification

Spec (in plaintext as accessibility for blind and low vision students) out: by Monday, January 16, 2023
Code due: by Monday, February 6, 2023 through bright.uvic.ca (also known as Brightspace or Bright)

Assignment objective:

In this programming assignment, you will use the STREAM socket (i.e., supported by TCP) in Python to create a Simple Web Server (SWS), mimicking nginx-light in PicoNet but only supporting limited functionalities as specified below, with select() supporting both persistent and non-persistent HTTP connections. Please note that you need to use select() non-blocking socket() to complete P1 as a preparation for P2.

Assignment schedule with regard to tutorials and labs:

There are two tutorials (T2 and T3) and two lab sessions (L2 and L3) dedicated to this assignment, with T1 and L1 as the warmup for your preparation for P1.

In T2, the tutorial instructor will go through the P1 specification, provide extra information and answer your questions, and will also give a simple design for your reference. In L2, the lab instructor will go through HTTP (both persistent and non-persistent, using nginx-light and wget) packet capture and analysis, and help students form their design for P1. If you have any questions about your SWS, please use nginx-light as a reference.

In T3, the tutorial instructor will check students P1 design, provide feedback and instruction on submission through Brightspace, and give a quick demo. In L3, the lab instructor will check students P1 implementation, capture and analyze SWS packets, and provide help if needed. Please attend all tutorials and labs in person.

Please follow our tutorial and lab schedule closely for this assignment, which ensures its success and smoothness.

SWS requirements:

SWS only supports the “GET /filename HTTP/1.0” command, and “Connection: keep-alive” and “Connection: close” request and response header when supporting persistent HTTP connection. The request header is terminated by an empty line known as “\r\n”, where “\r” indicates a carriage return and “\n” a (new) line feed.

If unsupported commands are received or in unrecognized format, SWS will respond “HTTP/1.0 400 Bad Request” and close the connection immediately.

If the file indicated by filename is inaccessible, SWS will return “HTTP/1.0 404 Not Found”. Such responses will be followed by the response header if any, an empty line, indicating the end of the response.

For successful requests, SWS will respond “HTTP/1.0 200 OK”, followed by the response header if any, an empty line indicating the end of the response header, and the content of the file.

In both “200” and “404” cases, if the client requests persistent connection, SWS will keep the connection open until closed by the client later or expired. Please refer to nginx-light for the expected behaviors of SWS.

How to run SWS:

On H2 in PicoNet, “python3 sws.py ip_address port_number”, where ip_address and port_number indicate where SWS binds its socket locally for incoming requests.

On H1 in PicoNet, “nc sws_ip_address sws_port_number” to connect to the remote SWS, and type “GET /sws.py HTTP/1.0” followed by “Connection: keep-alive” and an empty line to request the file sws.py from SWS (in this case, SWS shall keep the connection alive after sending back sws.py following an empty line after “Connection: keep-alive”, and wait for the next request from the same client through the same TCP connection, until the connection times out, i.e., “Connection: close”). If the client does not include “Connection: keep-alive” or does include “Connection: close” in its request, SWS will close the connection after serving the request.

For each served request, even if unsuccessfully, SWS will output a log line “time: client_ip:client_port request; response”, e.g., “Mon Jan 16 08:44:35 PST 2023: 192.168.1.100:54321 GET /sws.py HTTP/1.0; HTTP/1.0 200 OK”. Please follow the log format closely for marking purposes. Please note that SWS will keep waiting to serve more clients and can serve multiple concurrent clients by using select(), until interrupted by Ctrl-C.

How to test SWS:

Capture and analyze the interaction between SWS and its clients (nc, wget or even a regular web browser) with tcpdump and Wireshark. Your code will be evaluated in PicoLab as you have in ECS360.

What to submit:

sws.py source file, and the tcpdump files on R, showing the interaction between SWS and its clients, in both persistent (sws-persistent.cap) and non-persistent (sws-non-persistent.cap) connections.

When:

By Monday, February 6, 2023, through Brightspace -> assignments -> p1

Questions and answers:

In addition to associated tutorials and labs, use Teams -> assignments. Please be aware that the teaching team has no access to and is not responsible for the content and discussion on Discord.

Academic integrity:

This is an individual assignment and your submitted work shall be done by yourself alone. If you use any existing materials and/or libraries, you need to make explicit reference, so your work can be evaluated properly.

Appendix: A simple design for your reference