Project1: Web Server Programming

M2608.001200 Introduction to Data Communication Networks (2021 Fall)

Instructors: Prof. Kyunghan Lee
TAs: Kyungmin Bin, Taekyung Han
Due date: Nov. 7, 2021, 11:59 pm.

Notes:
- Be aware of plagiarism. You are allowed to use the eTL for QnAs, but do **NOT** discuss with your classmates **directly**.
- **Grading**: 100 pts
- In case you find out bugs, typos or issues in the given code, please report to the TAs by posting on the eTL. We will look into them and fix as soon as possible. Once fixed, we will upload the new version on the eTL by naming proper version number for the code and make an announcement on the eTL. Please pay attention to it and make sure you are using the latest version before submission.
- If you do not want other students to see your questions, you can always make them private.

Objective

The goal of this project is to implement a web server in three different ways: single-threaded, multi-threaded, and thread-pooled. These methods are known to provide substantial performance difference in serving multiple requests or clients. Understanding this performance difference by observing and monitoring the web server behaviors is a part of this project.

Program Specification

Skeleton code is provided in assignment tab of *eTL*. Copy the code to your Linux System (VM, MacOS, Ubuntu, etc.), and modify it to implement your own web server. Your web server should send requested files to the client through Hyper Text Transfer Protocol (HTTP).

Check the message format of HTTP (https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)~~.~~
Basically, you should fill up empty parts in the skeleton code to run this code successfully. This code is consisted of two functions: main() and respond()

main() function lets a socket bound to address and port (using bind() function) and listen on the socket (using listen() function). Then, it accepts a connection from a client (using accept() function) and sends the requested file to the client through HTTP message format. The server should continuously accept other clients after handling a request.

respond() function reads **a** request which contains **a** file path. In HTTP header, you can find requested file name. Then, you can take the file path (e.g, index.html, public/image1.jpg, bootstrap.js), read the files, attach the HTTP header to make a response message, and send the response message back to the client. You must explicitly implement the exception-handling part in **respond()** for the requests of resources or files that the server cannot handle. If you do not implement the exception-handling, you cannot get a full score. Refer https://en.wikipedia.org/wiki/List_of_HTTP_status_codes.

Implementation Guides

There are three ways to implement a web server.

1. Single-threaded Server

1) Wait for client request.

.

2) Process the request.

3) Repeat this cycle.

All of these steps are executed in a single thread.

2. Multi-threaded Server

Different from single-threaded server, multi-threaded server.
This link (https://www.geeksforgeeks.org/multithreading-c-2/) can be helpful to understand the implementation techniques.

1) Wait for client request.

2) When it accepts a new client connection, creates a new thread, and waits for another clients.

3) The new thread processes the new request.
→ Requests of clients can be processed concurrently.

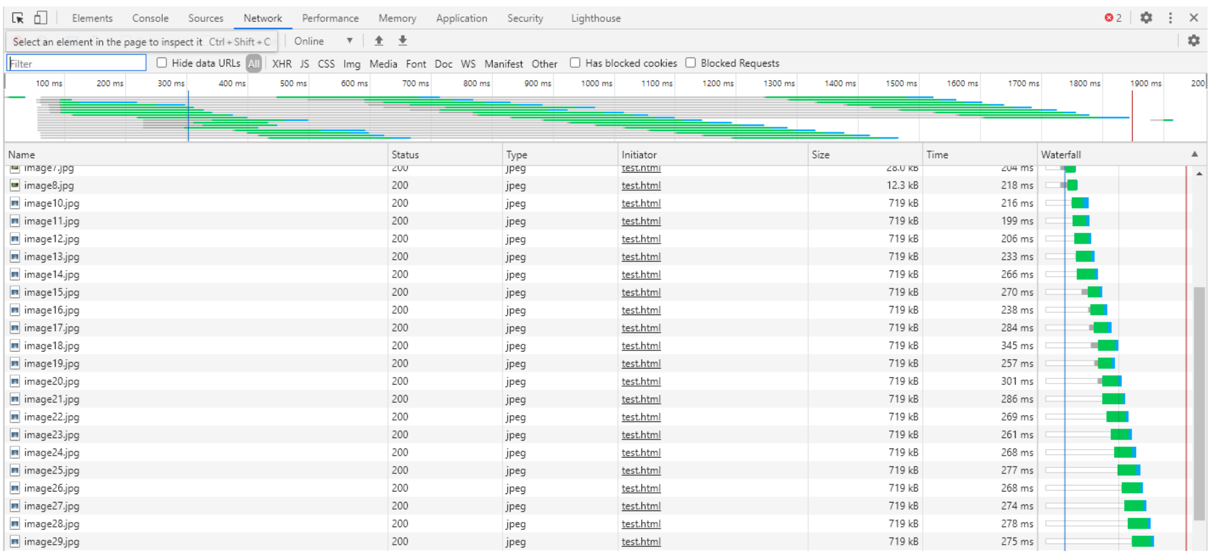
3. Thread-pooled Server

Web server creates threads very frequently, so it can consume lots of computing resources to create new threads. To solve this problem, thread-pooled server creates threads in advance, wake the thread and give task to the thread whenever it is needed. (https://en.wikipedia.org/wiki/Thread_pool)
The <threadpool/thpool.h> is provided in the skeleton code, so you can use this to implement thread-pooled server

This link (https://github.com/Pithikos/C-Thread-Pool) can be helpful to understand the implementation techniques.

Writing a Report

1) Open the Chrome developer tool (Ctrl + Shift + I)
2) Select 'network' tab
3) Load test.html
4) Capture the result
5) Describe what you find and understand from this monitoring tool in the "Report.docx" file.



Name	Status	Type	Initiator	Size	Time	Waterfall
image7.jpg	200	jpeg	test.html	28.0 kB	204 ms	
image8.jpg	200	jpeg	test.html	12.3 kB	218 ms	
image10.jpg	200	jpeg	test.html	719 kB	216 ms	
image11.jpg	200	jpeg	test.html	719 kB	199 ms	
image12.jpg	200	jpeg	test.html	719 kB	206 ms	
image13.jpg	200	jpeg	test.html	719 kB	233 ms	
image14.jpg	200	jpeg	test.html	719 kB	266 ms	
image15.jpg	200	jpeg	test.html	719 kB	270 ms	
image16.jpg	200	jpeg	test.html	719 kB	238 ms	
image17.jpg	200	jpeg	test.html	719 kB	284 ms	
image18.jpg	200	jpeg	test.html	719 kB	345 ms	
image19.jpg	200	jpeg	test.html	719 kB	257 ms	
image20.jpg	200	jpeg	test.html	719 kB	301 ms	
image21.jpg	200	jpeg	test.html	719 kB	286 ms	
image22.jpg	200	jpeg	test.html	719 kB	269 ms	
image23.jpg	200	jpeg	test.html	719 kB	261 ms	
image24.jpg	200	jpeg	test.html	719 kB	268 ms	
image25.jpg	200	jpeg	test.html	719 kB	277 ms	
image26.jpg	200	jpeg	test.html	719 kB	268 ms	
image27.jpg	200	jpeg	test.html	719 kB	274 ms	
image28.jpg	200	jpeg	test.html	719 kB	278 ms	
image29.jpg	200	jpeg	test.html	719 kB	275 ms	

Figure 1: Chrome developer tool example

Describe the characteristics of single-threaded, multi-threaded, and thread-pooled methods.
Analyze network downloading time and compare those three methods based on the graphs you obtain from monitoring tool.

Grading Policy

Grading policy is below:

Single-threaded Server : 40%

Multi-threaded Server : 30%

Thread-pooled Server : 10%

Report : 20%

You have to write comments in your codes for logically explaining what you did.
(If you don't write comments, you will get **-20% reduction**)
Project 1 will be **partially evaluated for several code blocks**. Even though your code can not operate properly, if your explains are logically correct, you can get partial points based on your comments in each function.

Compile and Usage

File name:

    server_single.c   server_multi.c   server_pool.c

Compile:

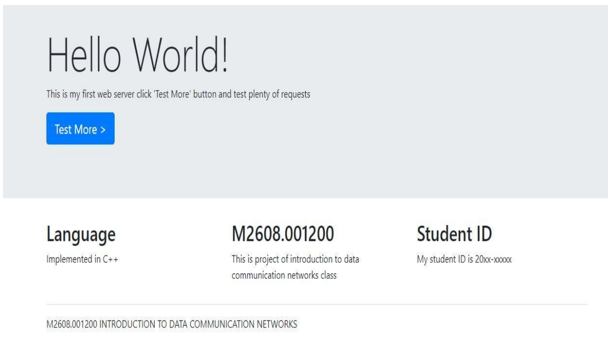
    make

Usage:

1) Check ip address of your VM (type in "ifconfig" in terminal).
2) Compile your code (make).
3) Run server (./server).
4) Open web browser in your desktop or smartphone.
5) Open link "http://IP:5000". (e.g., http://192.168.0.1:5000)

Output Specification

Before submitting your code, change "index.html" file to include your student ID.
Click "Test More" button and check bunch of image loading works well.


Hello World!

This is my first web server click 'Test More' button and test plenty of requests

Test More >

Language

Implemented in C++

M2608.001200

This is project of introduction to data communication networks class

Student ID

My student ID is 20xx-xxxxx

M2608.001200 INTRODUCTION TO DATA COMMUNICATION NETWORKS

Figure 2: index.html

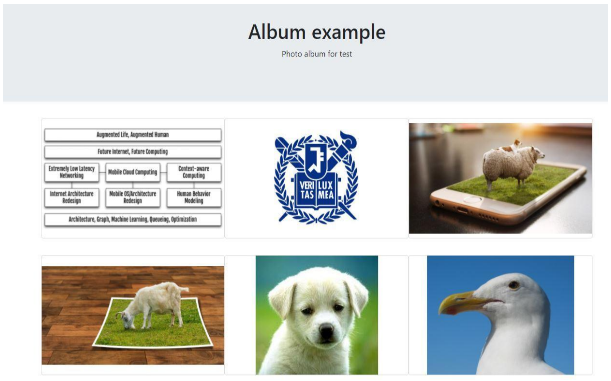

Album example
Photo album for test
Augmented Life, Augmented Human
Future Internet, Future Computing
Extremely Low Latency Networking
Mobile Cloud Computing
Context-aware Computing
Internet Architecture Redesign
Mobile OS/Architecture Redesign
Human Behavior Modeling
Architecture, Graph, Machine Learning, Queueing, Optimization
VERI TAS LUX MEA

Figure 3: test.html

Submission

Submit your codes and report to nxclab@gmail.com.
Title of the email should be in the format of "2021F-DCN-P1-StudentID".
Include your report and code to a folder "webserver_<student id>" and compress the folder.
Make the compressed zip file name as "webserver_<student id>".
You should submit only one compressed zip file.