Policies and Guidelines

- ***Assignments' deadlines come with a 72 hours grace period. Submitting the assignment during the grace period will be accepted with a 20% penalty. Submissions after the grace period will be accepted with a 50% penalty. No submission will be accepted after the last day of classes on Wednesday, May 7, 2025, at 11:59 pm.***
- Every project has required submission guidelines. Please read the submission requirements carefully. Any deviations from specifications on projects will result in point deductions or incomplete grades.
- ***Instructors will not 'fix' your code to see what works***. If your code does not run due to any type of errors, it is considered non-functioning.
- ***You cannot resubmit 'fixed' code after the deadline***, regardless of how small the error is.
- If you use the University resources for development and/or testing, then keep in mind that others in the University may need to use the same resources, so please use these resources wisely. You should also follow the rules and restrictions associated with using the University resources.
- ***Using any external/third-party library and/or framework to implement the project requirements is not permitted.***

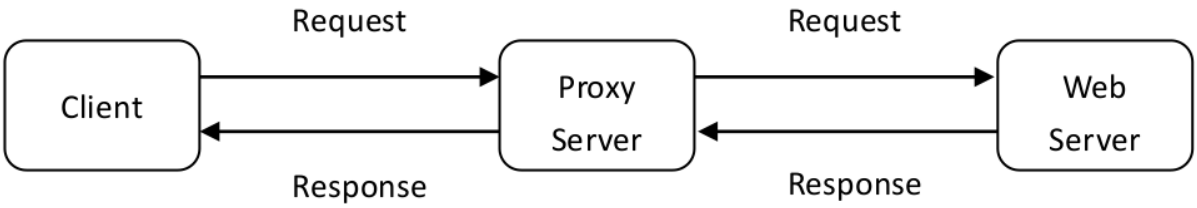
Academic Honesty Expectations and Violation Penalty

- ***The programming assignments are team assignments; each team cannot have more than two members***. Each team must do the vast majority of the work on its own. It is permissible to consult with other teams to ask general questions, help discover and fix specific bugs, and talk about high-level approaches in general terms. ***Giving or receiving answers or solution details from other teams is not permissible***.
- You may research online for additional resources; however, ***you may not use code explicitly written to solve the problem you have been given***. ***You may not have anyone else help you write the code or solve the problem***. You may use code snippets found online, providing that they are appropriately and clearly cited within your submitted code.
- ***If you use a distributed version control service such as GitHub to maintain your project, you should always set your project repository to private***. If you make your project repository public during the semester or even after the semester, it may cause a violation of the academic honesty policy if other students find and use your solution.

- If you are involved in plagiarism or cheating, you will receive a score of "0" for the involved project for the first offense. You will receive an "F" grade for the course and be reported to the university for any additional offense.
- Cheating and copying will **NOT** be tolerated. Anything submitted as a programming assignment must be the student's original work.
- ***The use of generative AI tools or apps for assignments, quizzes, and exams in this course, including tools like ChatGPT and other AI writing or coding assistants, is prohibited.*** The use of generative AI to complete any coursework may be considered use of an unauthorized aid, which is a form of cheating. This course policy is designed to promote your learning and intellectual development and to help you reach course learning outcomes.
- ***We reserve the right to use plagiarism detection tools to detect plagiarism in the programming assignments.***

Implementation

In this project, you will learn how web proxy servers work and one of their basic functionalities, forwarding. Generally, when the client makes a request, the request is sent to the web server. The web server then processes the request and sends back a response message to the requesting client. In order to improve the performance, we create a proxy server between the client and the web server. Now, both the request message sent by the client and the response message delivered by the web server pass through the proxy server. In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server. The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client, as shown in the figure below.



Part-1: Web Proxy Server

Extend your ***Project-2 multi-threaded web server (Part-2)*** implementation to implement and integrate a ***multi-threaded web proxy server***. ***Assume that caching is not enabled on the proxy server.*** In this case, the main functionality of the proxy server is to forward requests from the client to the web server and forward responses from the web server to the client.

Like the web server, the proxy server should create a separate thread to service each pair of client request and server response. When a client sends a connection request, the proxy server should establish a TCP connection with the client and place the connection on a new thread. The newly created thread on the proxy server should perform the following:

- Receive the client HTTP request.
- Establish a TCP connection with the web server.
- Forward the client request to the web server.
- Receive the web server response.
- Forward the response to the client.
- Close the web server TCP connection.
- Close the client TCP connection.
- Terminate the thread.

The proxy server should print the following information on the terminal when it forwards a request to the web server or forwards a response to the client:

```
proxy-forward,DESTINATION,THREAD-ID,TIMESTAMP
```

Where “DESTINATION” is either client or server, “THREAD-ID” is the ID of the proxy server thread processing the request/response, and “TIMESTAMP” is the time at which the proxy server forwarded the request/response.

Also, modify your web server implementation to print the following information on the terminal whenever it responds to a request from the proxy server:

```
server-response,STATUS-CODE,THREAD-ID,TIMESTAMP
```

Where the “STATUS-CODE” is the HTTP response code indicating the result of the request, “THREAD-ID” is the ID of the web server thread processing the request, and “TIMESTAMP” is the time at which the web server processed the request.

The web server and the proxy server should have a way to catch a termination signal and end their main thread safely. They should ensure all TCP connections are closed, and all threads are completed before terminating their main thread.

Testing the Proxy Server and Web Server Locally

Ensure the web server and proxy server files (source and executable) are each placed in a separate folder. Also, move the HTML/PDF/JPG files to the same folder in which the web server files are. Run the web server and the proxy server programs from two different terminals. Open a web browser and enter the URL link to request the HTML/PDF/JPG file using the proxy server address, i.e., the web browser only interacts with the proxy server, and the proxy server, in turn, is the one that interacts with the web server on behalf of the web browser. Try also to get a file not present on the web server. You should get a “404 Not Found” message.

Therefore, the web browser only needs to know the proxy server address. The proxy server, however, must know the web server address to forward the web browser request to it.

Submission

- Complete the "README.md" file. You may write “n/a” where applicable (bugs, references, etc.).
- Please submit the following files for Project-3 on Brightspace using the same file name and extension as here:
    - proxyserver.cc (or .c/.cpp/.h/.hpp).
    - webserver.cc (or .c/.cpp/.h/.hpp).
    - README.md
    - [Optional] To submit a makefile for your project, append a ".txt" extension to the file and then submit.
- You must submit your project before the deadline to be considered on time. Otherwise, it will be considered late even if your project was completely working before the deadline. ***Please note that on Brightspace, we maintain all your submissions but only grade your most recent submission.***

Grading Environment

All submissions will be tested and graded under the Ubuntu 24.04 LTS operating system. To run a submission, we will follow the instructions provided by the group in the README.md file. We will test each submission thoroughly to:

- Check the completeness and correctness of the implementation.
- Check if the implementation supports HTML, JPEG, and PDF files.
- Check the multi-threading is started correctly, works as expected, and terminates safely. We may use tools other than the web browser for testing, such as shell scripts.
- Check there are no compile-time, run-time, or memory leak errors using Valgrind.

Grading Rubric

Total: 100 points

- Part-1: 85 points.
- Submission: 15 points
    - README.md: 10 points.
    - Naming Requirements: 5 points.

- ***If Project-2 issues, if any, not resolved:*** -10 points.
- ***If submission didn't pass the plagiarism test(s):*** -100 points.
- ***Groups with two members:*** 3 additional points will be deducted ***"per"*** detected problem.