UTM Network Programming course for FAF-23 (Autumn 2025)

Books

I recommend reading “Computer Networking: a Top Down Approach”, skipping chapter 1. In chapter 2 you will learn everything you need for lab 1. You can find a copy in this folder.

Calendar

	Started	Deadline	Duration
Lab 1	Sep 30, 2025	Oct 14, 2025	2 weeks
Lab 2	Oct 14, 2025	Oct 21, 2025	1 weeks
Mid 1	Oct 27, 2025		
Lab 3	Oct 28, 2025	Nov 11, 2025	2 weeks
Lab 4	Nov 11, 2025	Nov 25, 2025	2 weeks

Labs

The PR labs will be interesting because you will write programs to network with your colleagues (e.g. query your friend’s HTTP server using your own client, host an online game server and play with your friends, etc.)

Rules:

- You must use Docker Compose
- You must use Python

How to present

Upload your project to GitHub and send me the URL in an email with the subject “PR lab 1” (artiom.balan@isa.utm.md). If you will be presenting later than the deadline, include the last commit hash so I can verify the time of the commit.

Update: Include a short report in the repository (markdown or a link to a Google doc). The report will be like a demo but with screenshots, so just include screenshots of

everything you would need to show in the demo: the commands you run, the outputs you see (browser, terminal) and write short descriptions where necessary.

You will present to me during seminars, defend your code, and answer a few theoretical questions.

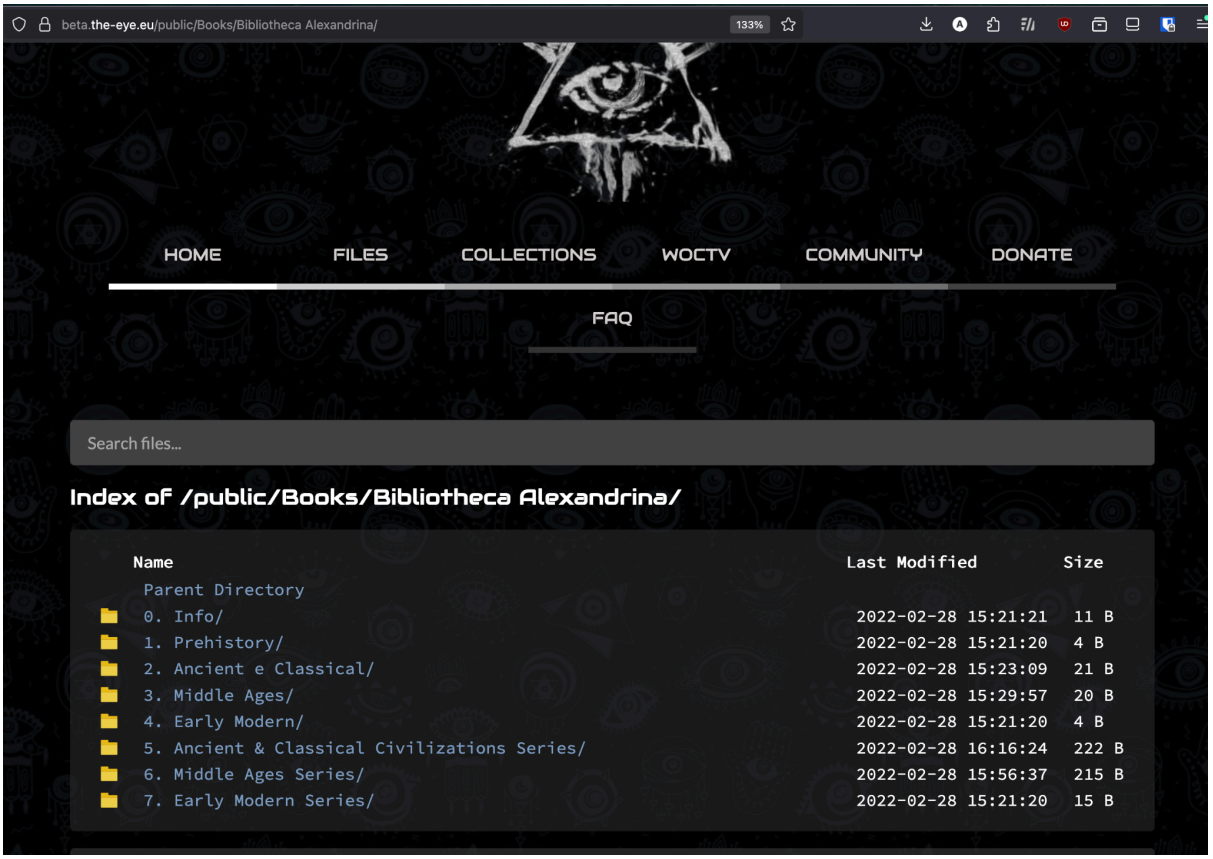
Docker

You must use Docker Compose for all your laboratory works so that I can run them on my computer.

The official Docker 101 tutorial is a good resource.

Lab 1: HTTP file server with TCP sockets

You will develop a simple HTTP file server like Python's http.server. You will use it to build a website that lets your friends browse your collection of PDFs, like this one:



It will handle one HTTP request at a time. The server program should take the directory to be served as a command-line argument.

Your web server should accept and parse the HTTP request, read the requested file from the directory, 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 (or has an unknown extension), the server should send an HTTP "404 Not Found" message back to the client. Update: your server should handle HTML, PNG and PDF file types.

Prepare a directory with content to be served. At the very least, it should contain an HTML file, a PNG image and a few PDF files. Reference the image in the HTML file (using <img>).

To get a 10, you need to do the following tasks as well:

1. Implement an HTTP client for your server - **2 points**

   Update: It will be a python script that takes as command-line arguments a URL and a directory to save files in, and acts depending on the file type:

   - HTML (page, directory listing): print the body of the response as-is
   - PNG, PDF: save the file in the specified directory

   Use the following format for the command arguments:
   client.py server_host server_port filename

2. Make the server work with nested directories - **2 points**:

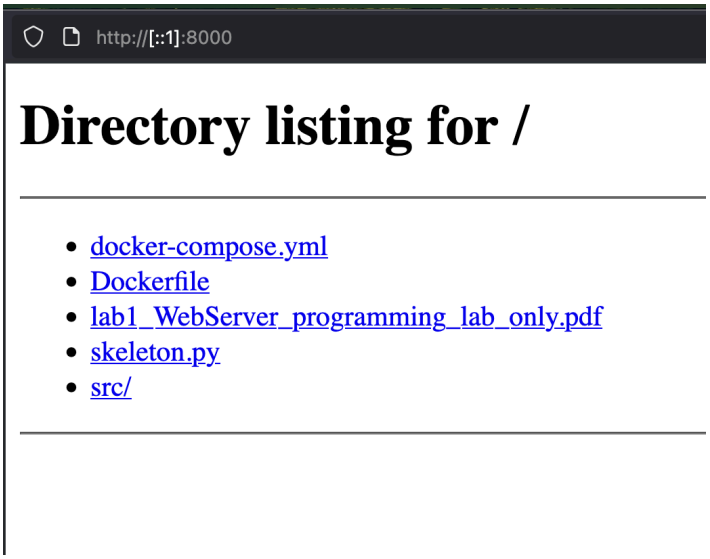
   If a directory path is requested, respond with a directory listing (a generated HTML page that displays the contents of the directory, with hyperlinks).

   Create a subdirectory in your content directory with a few PDF/PNG files inside it.

3. Browse the books on your friend's server (local network), have fun - **1 point**

   You can use your client to download a book from your friend's server right into your own website directory.

What the directory listing can look like (python's http.server response shown here):



Lab 2: Concurrent HTTP server

Work in progress

Make your server from lab 1 accept multiple connections at a time. Add a delay to the request handler to simulate work.

There are two ways to test it (can they use either?):

A. Using their own http client
B. Using a requests lib

Should they use thread pools or async/await? Or maybe they can pick either?

Additional exercises (optional, for better grade):

1. Query your friend's concurrent server, either with other people at the same time, or by making multiple concurrent requests (or better both). Compare the response time to the single-threaded server.

Lab 3

Work in progress

Lab 4: Multiplayer Game

Work in progress

You will implement the MIT 6.102 (2025) Memory Scramble lab.

A multiplayer game is a great lab for a PR course because it teaches important concepts in a fun way:

- Concurrency
- Implementing an HTTP API

Curriculum

Here is the list of topics that will be covered in this course:

1. Concurrency
    a. Threads
    b. Multiprocessing (maybe?)
    c. Promises (async/await)
2. Network programming
    a. Sockets, TCP, UDP
    b. Protocols, HTTP
    c. Serialization (JSON)
3. Git
4. Docker

Sources

It's better to make use of material made by professional educators than to come up with your own.

- "Computer Networking: a Top Down Approach" (Pearson,)
  - Has labs in python and slides for teaching.
  - Interactive animations
  - Sockets, web servers, protocols, proxies, etc.
  - Doesn't explain concurrency.
  - Lab 1: HTTP server with TCP sockets
    - Additional 1: make it multithreaded
    - Additional 2: write HTTP client
  - Lab 2: UDP client + server, timeouts
  - Lab 3: implement basic SMTP protocol
  - Lab 4: implement Ping using raw sockets (ICMP)
    - Lab 4.2: implement traceroute using ICMP
  - Lab 5: HTTP proxy server
  - Lab 6: video streaming, implement RTSP with sockets
- University Course MIT 6.102 "Software Construction" (2025)
  - Concurrency (high-level):
    - 15: Promises
    - 16: Mutual Exclusion
    - Message passing
  - Low-level concurrency in older course: MIT 6.005 "Software Construction" (2016):
    - Thread safety
    - Locks & synchronization
- The definitive Glossary of concurrency concepts by slikts
- A terse and elegant overview of concurrency (independence) and concurrency paradigms without observable nondeterminism: Programming Paradigms for Dummies: What Every Programmer Should Know by Peter Van Roy
- A comprehensive book on the concepts of concurrency: "The Art of Multiprocessor Programming" (2008) (multiprocessor = multi-core):
  - Theoretical (Mutual Exclusion)
  - Practical (Locks, Monitors)