Course	COMP 4981
Program	Diploma of Technology, Datacommunications & Internetworking
Term	January 2025

- This is a pair programming assignment.

Objective

- Develop a comprehensive understanding of network programming by implementing an advanced multi-process HTTP server using POSIX sockets in C.
- The server must use pre-forking, support dynamic updates via shared libraries, and handle HTTP POST requests with persistent data storage in an ndbm database.

Learning Outcomes

- Apply POSIX socket APIs to create high-performance networked applications.
- Implement a pre-forking server model to handle concurrent connections efficiently.
- Dynamically load and update HTTP handling logic using shared libraries.
- Support HTTP POST requests and store data persistently in an ndbm database.
- Ensure compatibility across Linux, FreeBSD, and macOS platforms.

Assignment Details

- You must develop an HTTP server capable of handling multiple client connections concurrently using a pre-forked worker model.
- The server must dynamically load a shared library for request handling, allowing updates to be applied without restarting the server.
- Additionally, it must support HTTP GET, HEAD, and POST requests, storing data in an ndbm database.

Requirements

Server Functionality

- Accept multiple client connections using TCP sockets.
- Handle HTTP GET, HEAD, and POST requests.
- Serve requested files from a designated directory for GET and HEAD requests.
- Store data from POST requests into an ndbm database.

- Respond with appropriate HTTP status codes for successful requests, unsupported methods, and missing files.
- The main server process is responsible for closing client connections, not the forked handlers.

Concurrency & Process Management

- Pre-fork a configurable number of worker processes to handle client requests.
- Monitor child processes and restart them if they crash.
- Ensure proper cleanup of worker processes on shutdown.

Dynamic Library Loading

- Implement HTTP request handling in a shared library (.so).
- Before serving each client, check if a newer shared library version is available.
- If a newer version is found, reload it dynamically using dlsym().

Data Storage with ndbm

- Store data received via POST requests in an ndbm database.
- Implement a separate program that can query and display stored data.

Error Handling

- Implement robust error handling for system calls and memory management.
- Ensure the server can recover gracefully from failures.

Compatibility

- Ensure your code compiles and runs on Linux, FreeBSD, and macOS.

Constraints

- Use only POSIX socket APIs and standard C libraries.
- Do not use external libraries for networking, concurrency, or HTTP parsing.
- Ensure the program works across all specified operating systems.

Resources

- Making a simple HTTP webserver in C - bruinsslot.jp
- Web server test suite

Submission

- Ensure your submission meets all the guidelines, including formatting, file type, and submission.
- Follow the AI usage guidelines.
- Be aware of the late submission policy to avoid losing marks.
- ***Note: Please strictly adhere to the submission requirements to ensure you don't lose any marks.***

Evaluation

Topic	Value
Design	25
Testing	25
Implementation	50
Total	100%

Hints

- Start with a single-process HTTP server handling GET and HEAD requests.
- Introduce pre-forking and ensure worker processes handle requests correctly.
- Implement dynamic shared library loading to support updates.
- Add POST request handling and integrate the ndbm database.
- Test thoroughly across Linux, FreeBSD, and macOS.
- Use logging to debug and understand server behaviour.