Course	COMP 7005
Program	Bachelor of Science in Applied Computer Science
Term	April 2025

- This is an individual programming assignment.

Objective

- Develop a multi-client server application using network sockets for inter-process communication.
- This assignment focuses on concurrent server design using I/O multiplexing, file transfer, and Vigenère cipher decryption.

Learning Outcomes

- Implement a server capable of handling multiple client connections concurrently.
- Use network sockets for client-server communication.
- Apply Vigenère cipher decryption to transform received data.
- Gain experience in designing and testing inter-process communication programs.

Assignment Details

Requirements

You will write two programs.

Client

- The client program must:
    - Accept the following inputs as command-line arguments:
        - The IP address and port number of the server.
        - The name of the encoded file to send to the server.
        - The keyword for Vigenère cipher decryption.
    - Read the contents of the specified encoded file.
    - Connect to the server via a network socket using the provided IP address and port number.
    - Send the keyword and the file's encoded content to the server.
    - Receive the decoded file content from the server and print it to the terminal.

Server

- The server program must:
    - Accept the following inputs as command-line arguments:
        - The IP address and port number to bind to.
        - A configurable delay (in seconds) before encryption.
        - A configurable delay (in seconds) after encryption and before sending data back.
    - Use select() or poll() to handle multiple client connections concurrently.
    - For each client:
        - Receive the keyword and file content.
        - Print that a connection has received data.
        - Print that it is about to encrypt, then wait for the configured pre-encryption delay.
        - Encrypt the data using the Vigenère cipher and print the encryption completed.
        - Wait for the configured post-encryption delay.
        - Print that it is sending the data back, and send the encrypted result.
    - Ensure proper error handling, cleanup of closed connections, and graceful exit.

Constraints

- You may use any language you like.
- The program must run on a UNIX-like Operating System (e.g., Linux or macOS).
- Use network sockets exclusively for client-server communication.
- Implement proper error handling for socket operations.
- Implement cleanup mechanisms to handle closed or failed connections.
- Implement the Vigenère cipher decryption manually:
    - Assign numerical values to letters (A = 0, B = 1, ..., Z = 25).
    - Use modulo 26 arithmetic for wraparound behaviour.
    - Decrypt only alphabetic characters, preserving non-alphabetic characters as-is.
- Do not use higher-level libraries for socket programming.
- Do not use external encryption libraries; implement the Vigenère cipher algorithm from scratch.

Resources

- Man Pages: man 2 socket, man 3 read, man 3 write, man 3 getopt.
- Vigenère Cipher Explanation: Wikipedia.
- Code samples from your course materials.

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
Correct implementation of the client program	15%
Correct implementation of the server program	15%
Proper use of Multiplexing	20%
Design	20%
Testing	30%
Total	100%

Hints

- Be sure to test with a large file that causes the encoded input to be split across multiple reads/writes.
- Use temporary files or debug logs to monitor data at each stage.
- Print messages at all key stages: when data is received, before delay, after decryption, and before sending.
- Validate all command-line inputs.
- Use 127.0.0.1 for testing on a single machine; test concurrency using multiple terminals or machines.
- Use usleep() or sleep() in C to implement delays.