Client-Server Text Encryption Application

Overview

This C++-based application implements a client-server architecture for encrypting and decrypting text using a Caesar cipher. The server supports multiple parallel client connections using threads. The client sends requests to the server to either encrypt or decrypt text. The server processes the request and returns the result to the client.

Features

- Encryption and Decryption Commands:
  - encr <text>: Encrypts the provided text using Caesar cipher.
  - decr <text>: Decrypts the provided text using Caesar cipher.
- Multithreading:
  - The server handles multiple client connections in parallel using detached threads.
- Networking:
  - Based on the Winsock2 library for TCP communication.

Implementation Overview

- Language: C++
- Networking Library: Winsock2
- Multithreading Library: C++ Standard Library (std::thread, std::mutex)
- Cipher: Caesar Cipher

Workflow

Client Workflow

1. The client establishes a connection to the server.
2. The user inputs commands:
   - encr <text> to encrypt text.
   - decr <text> to decrypt text.
3. The client sends the command to the server.

4. The client waits for the server’s response and displays the result.
5. The process repeats until the user exits by typing exit.

Server Workflow

1. The server listens for incoming client connections on a specified port.
2. For each new connection, the server spawns a thread to handle the client.
3. The server processes the client’s command:
    - Encrypts or decrypts the text using the Caesar cipher.
    - Sends the result back to the client.
4. The server continues to handle multiple clients simultaneously.

Caesar Cipher Algorithm

Encryption

- Each character in the input text is shifted backward by a fixed number of positions (KEY).
- If the character goes below 'A' (uppercase) or 'a' (lowercase), it wraps around to 'Z' or 'z', respectively.

Decryption

- Each character in the input text is shifted forward by the same fixed number of positions (KEY).
- If the character exceeds 'Z' (uppercase) or 'z' (lowercase), it wraps around to 'A' or 'a', respectively.

Commands and Examples

Encrypt Command

Client Input: encr Hello World

Server Response: Ebiil Tloia

Decrypt Command

Client Input: decr Ebiil Tloia

Server Response: Hello World

Invalid Command

Client Input: xyz Hello

Server Response: Operation not specified!

Code Details

Server Implementation

1. Setup Winsock:
    - Initialize Winsock using WSAStartup.
    - Create a socket and bind it to the specified port.
    - Listen for incoming connections.
2. Multithreaded Client Handling:
    - For each accepted client connection, spawn a new thread to handle the client.
    - Each thread performs the following:
        - Reads data sent by the client.
        - Parses the command and processes the text using the Caesar cipher.
        - Sends the result back to the client.
3. Cipher Functions:
    - encryptWord: Encrypts a single word by shifting characters backward.
    - decryptWord: Decrypts a single word by shifting characters forward.
    - splitStringBySpaces: Splits the input text into words for individual processing.
    - CeserEncrypt and CeserDecrypt: Apply encryption or decryption to the entire input text.

Client Implementation

1. Setup Winsock:
    - Initialize Winsock using WSAStartup.
    - Create a socket and connect it to the server.
2. User Interaction:

- Read user input and send it to the server.
- Receive the server’s response and display it.
- Repeat until the user exits by typing exit.

3. Communication:
   - Send commands to the server.
   - Handle server disconnection gracefully.

Example Usage

Server

Server is listening on port 55555...

Connection established with client.

Received: encr Hello World

Sent: Ebiil Tloia

Client

Connected to server. Type text to send (type 'exit' to quit):

Hello to Caesar Cipher. Choose what you want: to encrypt ("encr") or decrypt ("decr").

> encr Hello World

Server response: Ebiil Tloia

> decr Ebiil Tloia

Server response: Hello World

Compile and Run

Server

1. Compile the server code:

```
g++ -o server server.cpp -lws2_32
```

2. Run the server:

```
./server
```

Client

1. Compile the client code:

```
g++ -o client client.cpp -lws2_32
```

2. Run the client:

```
./client
```

Conclusion

This project demonstrates a simple yet effective implementation of a client-server architecture for text encryption and decryption. The multithreaded server can handle multiple clients concurrently, showcasing the use of parallelism in network programming. The Caesar cipher provides a basic example of cryptographic transformations.