**Socket Client-Server Communication**

This project demonstrates a simple client-server communication model using TCP sockets in C. The client sends messages to the server, and the server processes these messages and responds. The server can handle multiple clients concurrently using process forking.

**Getting Started**

Follow the instructions below to set up and run the client and server programs.

**Prerequisites**

Ensure you have a C compiler (such as gcc) and necessary libraries for socket programming. This project is intended for Unix-like operating systems.

**Compilation**

To compile the client and server programs, run the following commands:

bash

Copy code

gcc -o socket-server socket-server.c

gcc -o socket-client socket-client.c

**Usage**

1. **Start the Server**: Run the server program with the desired IP address and port number. The server will listen for incoming connections on this address and port.

bash

Copy code

./socket-server <IP\_ADDRESS> <PORT>

Example:

bash

Copy code

./socket-server 0.0.0.0 12345

1. **Start the Client**: Run the client program with the server's IP address and port number. The client will connect to the server and enable you to send messages.

bash

Copy code

./socket-client <SERVER\_IP\_ADDRESS> <PORT>

Example:

bash

Copy code

./socket-client 127.0.0.1 12345

1. **Interact with the Server**:
   * Type messages in the client terminal to send them to the server.
   * The server will respond to each message.
   * To disconnect, type "exit" in the client terminal.

**Code Overview**

**socket-client.c**

The client program performs the following steps:

1. Initializes and parses command-line arguments for the server IP address and port number.
2. Creates a socket using socket().
3. Resolves the server address using gethostbyname().
4. Establishes a connection to the server using connect().
5. Enters a loop to:
   * Read user input.
   * Send the input to the server.
   * Receive and display the server's response.
   * Measure and display the round-trip time for each message.
   * Disconnect if the message "exit" is sent.

**socket-server.c**

The server program performs the following steps:

1. Initializes and parses command-line arguments for the IP address and port number to bind to.
2. Creates a socket using socket().
3. Binds the socket to the specified IP address and port using bind().
4. Listens for incoming connections using listen().
5. Enters an infinite loop to:
   * Accept incoming client connections using accept().
   * Fork a new process to handle each client.
   * In the child process, enter a loop to:
     + Read messages from the client.
     + Check for the "bye" message to disconnect.
     + Send a response to the client ("accepted!\n" for other messages).
6. Closes the client and server sockets before exiting.

**License**

This project is licensed under the MIT License. See the LICENSE file for more details.

Feel free to open issues or submit pull requests if you have any questions or suggestions. Enjoy exploring and modifying the code!