Homework 3: Chat Client/Server in Linux

Submitted by: Sagi Ber (208276188), Rotem Refaeli (322281775), Roy Porat (205947690)

Overview

This documentation covers the implementation details of the server.c and client.c files which are part of a client/server chat in Linux using the TCP protocol. The server listens for connections from clients, handles their requests, and sends responses. The client connects to the server, sends requests, and processes the responses.

Api

- **#include <stdio.h>:** provides functions for input/output operations.
- **#include <stdlib.h>:** Provides general-purpose standard library functions for memory management, random numbers, conversions, etc.
- **#include <string.h>:** Provides functions for manipulating arrays of characters (strings), including operations such as strlen, strcpy, strncpy, and memcmp.
- **#include <unistd.h>:** Provides access to the POSIX operating system API, including system calls like read, write, close, and fork.
- **#include <sys/types.h>:** Defines numerous data types used in system calls. These types are necessary for various functions and system calls, such as those that involve file system operations or modifying process attributes.
- **#include <sys/socket.h>:** Provides functions for socket programming, allowing for the creation of network connections and communication. This header includes functions like socket, bind, listen, and accept.
- **#include <netinet/in.h>:** Contains constants and structures needed for internet domain addresses. This header defines structures such as sockaddr_in (used for IP addresses) and constants like INADDR_ANY.
- **#include <netdb.h>:** Defines functions and data structures for network database operations, like translating a hostname to an IP address, thus involving functions such as gethostbyname and gethostbyaddr.
- **#include <pthread.h>:** Provides functions for managing POSIX threads (multi-threading).
- **#include <arpa/inet.h>:** Provides functions for manipulating numeric IP addresses. It includes functions that convert Internet addresses in network byte order to and from their string representations.

#define Constants

Name	Value	Purpose
BUFFER_SIZE	256	Buffer size for massage
MAX_CLIENTS	16	The maximum number of clients allowed

Code Structure

Server (server.c**)**:

1. **Initialization**:
   - Setup socket and network communication parameters.
   - Bind to the specified port and start listening for clients.
2. **Connection Handling:**
   - Accept new connections.
   - Read data from the socket.
   - Process requests and send responses.
3. **Cleanup:**
   - Close sockets and release resources.

Client (client.c**)**:

1. **Setup:**
   - Create a socket and connect to the server.
2. **Communication:**
   - Send requests to the server (private/broadcast massage).
   - Receive and process responses from the server.
3. **Termination:**
   - Close the connection and cleanup resources.

Compilation and Debugging

All input files and command-line arguments are assumed to be valid. The code was compiled using the GCC compiler with strict warning flags enabled. Additionally, the program was thoroughly debugged using GDB to trace execution flow, identify segmentation faults, and analyze thread behaviour, ensuring correctness and reliability.

Summary

This program demonstrates fundamental network programming concepts in C using sockets. It provides a basic framework for creating client-server applications that can be extended for various network-based services. We've learned how to send/recivev a private massage, to synchronize incoming massages without a delay, and to broadcast a massage to all the clients that connected in the server.