UDP chat with DNS lookup

Project Team

- Amisha Nath(1NT21EC010)
- Anarghya Amarnath(1NT21EC015)
- Pranjali Hebbar (1NT21EC099)
- Saagar E.P (1NT21EC122)

Objective

The objective of this project is to develop a UDP-based client-server application that can handle domain-to-IP and IP-to-domain queries. The server reads a DNS file and responds to client requests accordingly, demonstrating the basics of socket programming, file handling, and inter-process communication.

Tools:

- **Windows Subsystem for Linux (WSL)**: To provide a Linux environment on Windows.
- **Ubuntu**: A Linux distribution used within WSL.
- **GNU Compiler Collection (GCC)**: To compile C program.
- **Nano**: A text editor for creating and editing source code.

Work Flow/Execution:

1. Setup and Installation:

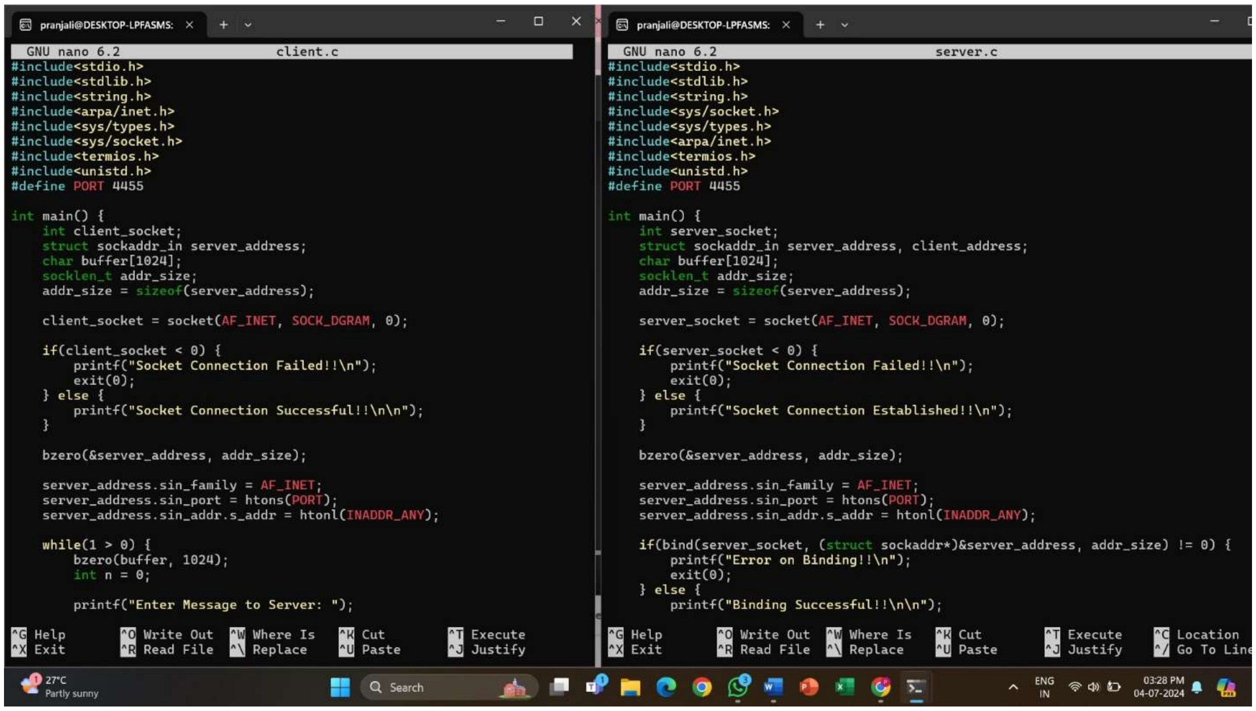
- **Enable WSL**: Open PowerShell as an administrator and execute `wsl –install`. Restart the computer if prompted.
- **Launch Ubuntu**: Open Ubuntu from the Start menu and complete the setup.
- **Install Build Tools**: In the Ubuntu terminal, update the package list with `sudo apt update` and install build-essential using `sudo apt install build-essential`.

2. Create and Edit Source Files:

- **Client Code**: Create a file `client.c` using `nano client.c`. Type the client code, save, and exit.

- **Server Code**: Create a file `server.c` using `nano server.c`. Type the server code, save, and exit.

- **DNS File**: Create `DNS.txt` using `nano DNS.txt`, add sample domain-to-IP mappings, save, and exit.



```
  GNU nano 6.2                      client.c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<termios.h>
#include<unistd.h>
#define PORT 4455

int main() {
    int client_socket;
    struct sockaddr_in server_address;
    char buffer[1024];
    socklen_t addr_size;
    addr_size = sizeof(server_address);

    client_socket = socket(AF_INET, SOCK_DGRAM, 0);

    if(client_socket < 0) {
        printf("Socket Connection Failed!!\n");
        exit(0);
    } else {
        printf("Socket Connection Successful!!\n\n");
    }

    bzero(&server_address, addr_size);

    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(PORT);
    server_address.sin_addr.s_addr = htonl(INADDR_ANY);

    while(1 > 0) {
        bzero(buffer, 1024);
        int n = 0;

        printf("Enter Message to Server: ");
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

```
  GNU nano 6.2                                  server.c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<termios.h>
#include<unistd.h>
#define PORT 4455

int main() {
    int server_socket;
    struct sockaddr_in server_address, client_address;
    char buffer[1024];
    socklen_t addr_size;
    addr_size = sizeof(server_address);

    server_socket = socket(AF_INET, SOCK_DGRAM, 0);

    if(server_socket < 0) {
        printf("Socket Connection Failed!!\n");
        exit(0);
    } else {
        printf("Socket Connection Established!!\n");
    }

    bzero(&server_address, addr_size);

    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(PORT);
    server_address.sin_addr.s_addr = htonl(INADDR_ANY);

    if(bind(server_socket, (struct sockaddr*)&server_address, addr_size) != 0) {
        printf("Error on Binding!!\n");
        exit(0);
    } else {
        printf("Binding Successful!!\n\n");
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

3. Compile the Source Files:

- **Compile Client**: Execute `gcc -o client client.c` in the terminal.

- **Compile Server**: Execute `gcc -o server server.c` in the terminal.

4. **Run the Application:**

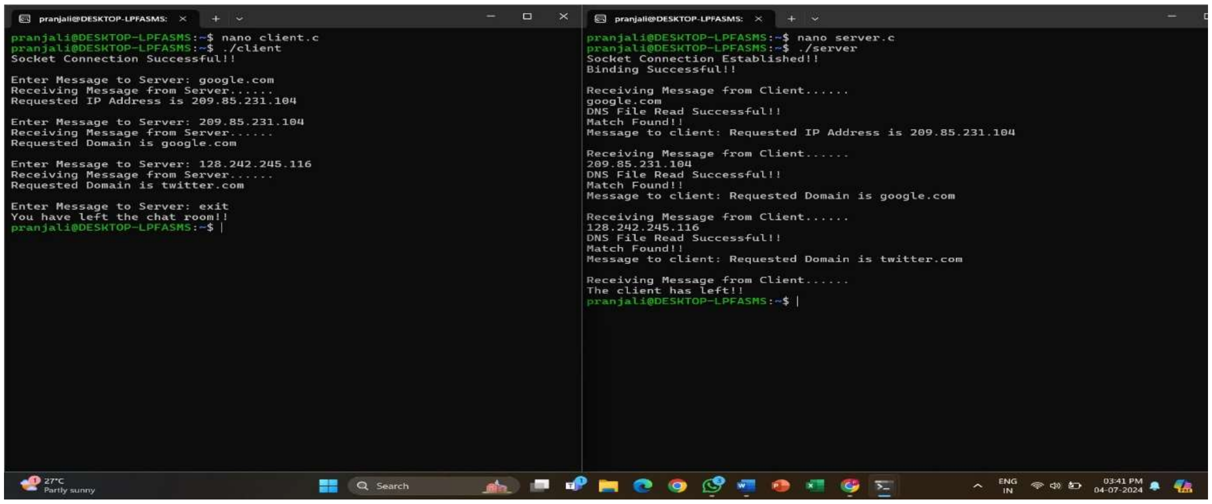
- **Start the Server**: Open a new terminal window and run `./server`.

- **Start the Client**: In the original terminal window, run `./client` and interact by entering domain/IP queries.

Results

- **Client-Side**: The client successfully sends queries to the server and receives responses. Typing `example.com` returns its corresponding IP, and vice versa.

- **Server-Side**: The server correctly processes the client's requests, searches the `DNS.txt` file for matches, and sends the appropriate responses back to the client. Both client and server terminate upon receiving the `exit` message.



```
pranjali@DESKTOP-LPFASMS:~$ nano client.c
pranjali@DESKTOP-LPFASMS:~$ ./client
Socket Connection Successful!!

Enter Message to Server: google.com
Receiving Message from Server......
Requested IP Address is 209.85.231.104

Enter Message to Server: 209.85.231.104
Receiving Message from Server......
Requested Domain is google.com

Enter Message to Server: 128.242.245.116
Receiving Message from Server......
Requested Domain is twitter.com

Enter Message to Server: exit
You have left the chat room!!
pranjali@DESKTOP-LPFASMS:~$ |
```

```
pranjali@DESKTOP-LPFASMS:~$ nano server.c
pranjali@DESKTOP-LPFASMS:~$ ./server
Socket Connection Established!!
Binding Successful!!

Receiving Message from Client......
google.com
DNS File Read Successful!!
Match Found!!
Message to client: Requested IP Address is 209.85.231.104

Receiving Message from Client......
209.85.231.104
DNS File Read Successful!!
Match Found!!
Message to client: Requested Domain is google.com

Receiving Message from Client......
128.242.245.116
DNS File Read Successful!!
Match Found!!
Message to client: Requested Domain is twitter.com

Receiving Message from Client......
The client has left!!
pranjali@DESKTOP-LPFASMS:~$ |
```

Conclusion

The UDP Client-Server Application demonstrates basic UDP socket programming and file handling in a Linux environment. By following the outlined steps, we successfully created a functional application that handles domain and IP address queries. This project serves as an example to network programming concepts and practical implementation using the C programming language and Linux tools.