Exercise 2: UDP Sensor Client-Server

Objective

Learn how to use UDP sockets in C++ to send and receive simple sensor data.

Task

You will write two C++ programs:

1. Sensor Client

- Sends a UDP packet every second.
- The packet contains a simple message like: `Temperature: 23°C`.

2. Server

- Listens on a UDP port.
- Prints every message received.

Optional Challenge

- Add a simple checksum to the message.
- Make the client resend the message if no acknowledgment is received within 1 second.

Starter Code

sensor_client.cpp (Stub)

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sockfd;
    struct sockaddr_in server_addr;

    // TODO: Create UDP socket

    // TODO: Fill in server information (IP, Port)
```

```
    while (true) {
        // TODO: Create message

        // TODO: Send message to server

        // TODO: Sleep for 1 second
    }

    // TODO: Close socket

    return 0;
}
```

sensor_server.cpp (Stub)

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);

    // TODO: Create UDP socket

    // TODO: Bind to a port

    while (true) {
        char buffer[1024];
        memset(buffer, 0, sizeof(buffer));

        // TODO: Receive message from client

        // TODO: Print received message

        // TODO (Optional): Send acknowledgment
    }

    // TODO: Close socket

    return 0;
}
```

Learning Goals

- Understand how UDP works at the socket level.
- Practice sending and receiving data over the network.
- Realize the stateless and unreliable nature of UDP.

Hint: Look up these functions:

- `socket()`
- `sendto()`
- `recvfrom()`
- `bind()`
- `close()`