```
#include <iostream>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <cstring>
#include <thread>

#define PORT 8080

void receiveMessages(int sock) {
    char buffer[1024] = {0};
    while (true) {
        memset(buffer, 0, 1024);
        int valread = read(sock, buffer, 1024);
        if (valread <= 0) break;
        std::cout << "Server: " << buffer << std::endl;
    }
    close(sock);
}

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[1024] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        std::cout << "Socket creation error" << std::endl;
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        std::cout << "Invalid address/ Address not supported" << std::endl;
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        std::cout << "Connection Failed" << std::endl;
        return -1;
    }
```

```
    std::thread recvThread(receiveMessages, sock);
    recvThread.detach();

    while (true) {
        std::string clientMessage;
        std::cout << "You: ";
        std::getline(std::cin, clientMessage);

        send(sock, clientMessage.c_str(), clientMessage.length(), 0);
    }

    close(sock);
    return 0;
}
```

Server code

```
#include <iostream>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <cstring>
#include <thread>
#include <signal.h>

#define PORT 8080

// Function to handle each client connection
void clientHandler(int clientSocket);

int server_fd;

// Signal handler for graceful shutdown
void signalHandler(int signum) {
    std::cout << "Interrupt signal (" << signum << ") received.\n";
    close(server_fd);
    exit(signum);
}

int main() {
```

```
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);

    // Register signal handler for SIGINT
    signal(SIGINT, signalHandler);

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt))) {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    std::cout << "Server listening on port " << PORT << std::endl;

    while (true) {
        int new_socket;
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen)) <
0) {
            perror("accept");
            exit(EXIT_FAILURE);
        }

        // Create a thread to handle the new client
```

```
        std::thread(clientHandler, new_socket).detach();
    }

    close(server_fd);
    return 0;
}

// Function to handle each client connection
void clientHandler(int clientSocket) {
    char buffer[1024] = {0};
    const char *hello = "Hello from server";

    // Read and print message from client
    read(clientSocket, buffer, 1024);
    std::cout << "Message from client: " << buffer << std::endl;

    // Send hello message back to client
    send(clientSocket, hello, strlen(hello), 0);
    std::cout << "Hello message sent\n";

    // Close the client socket
    close(clientSocket);
}
```

Client code

```
#include <iostream>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <cstring>
#include <signal.h>
#include <thread>

#define PORT 8080

int sock;
```

```
void signalHandler(int signum) {
    std::cout << "\nSignal (" << signum << ") received. Informing server and closing client." <<
std::endl;
    const char *goodbye = "Client is shutting down";
    send(sock, goodbye, strlen(goodbye), 0);
    close(sock);
    exit(signum);
}

void receiveMessages() {
    char buffer[1024] = {0};
    while (true) {
        memset(buffer, 0, 1024);
        int valread = read(sock, buffer, 1024);
        if (valread <= 0) break;
        std::cout << "Server: " << buffer << std::endl;
    }
    close(sock);
}

int main() {
    struct sockaddr_in serv_addr;
    const char *hello = "Hello from client";

    signal(SIGINT, signalHandler);
    signal(SIGTERM, signalHandler);

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        std::cout << "Socket creation error" << std::endl;
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        std::cout << "Invalid address/ Address not supported" << std::endl;
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        std::cout << "Connection Failed" << std::endl;
        return -1;
```

```
    }

    std::thread recvThread(receiveMessages);
    recvThread.detach();

    while (true) {
        std::string clientMessage;
        std::cout << "You: ";
        std::getline(std::cin, clientMessage);
        send(sock, clientMessage.c_str(), clientMessage.length(), 0);
    }

    close(sock);
    return 0;
}
```