

TCP Socket을 이용한 Chat Program 작성

1. 채팅 프로그램 실행 결과

```
machine1 - VMware Workstation 17 Player
Player
현재 활동
터미널
3월 21일 22:19
EN
root
root@ub1-virtual-machine: ~/바탕화면/server
root@ub1-virtual-machine:~/바탕화면# cd server
root@ub1-virtual-machine:~/바탕화면/server# gcc -o server server.c
root@ub1-virtual-machine:~/바탕화면/server# ./server
Server is listening on port 12345
Client connected
Received message: Hi
Enter message: Hi. You are a client, right?
Received message: Yes. It's a test for 1:1 chatting.
Enter message: Confirmed.
```

```
machine2 - VMware Workstation 17 Player
Player
현재 활동
터미널
3월 21일 22:19
EN
root
root@ub2-virtual-machine: ~/바탕화면/client
root@ub2-virtual-machine:~/바탕화면# cd client
root@ub2-virtual-machine:~/바탕화면/client# gcc -o client client.c
root@ub2-virtual-machine:~/바탕화면/client# ./client
Connected to server
Enter message: Hi
Server echoed: Hi. You are a client, right?
Enter message: Yes. It's a test for 1:1 chatting.
Server echoed: Confirmed.
Enter message:
```

2. Source Code

(1) Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 12345
#define BUFFER_SIZE 1024
int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];
    // Create socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket == -1) {
        perror("Error creating socket");
        exit(EXIT_FAILURE);
    }
    // Bind
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(PORT);
    if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) ==
-1) {
        perror("Error binding");
        exit(EXIT_FAILURE);
    }
    // Listen
    if (listen(server_socket, 5) == -1) {
        perror("Error listening");
        exit(EXIT_FAILURE);
    }
    printf("Server is listening on port %d\n", PORT);
    // Accept connection
    client_socket = accept(server_socket, (struct sockaddr *)&client_addr,
&addr_len);
    if (client_socket == -1) {
        perror("Error accepting connection");
        exit(EXIT_FAILURE);
    }
}
```

```

}
printf("Client connected\n");
while (1) {
    // Receive message from client
    ssize_t recv_len = recv(client_socket, buffer, BUFFER_SIZE, 0);
    if (recv_len <= 0) {
        perror("Error receiving data");
        break;
    }
    buffer[recv_len] = '\0'; // Null-terminate the received data
    printf("Received message: %s\n", buffer);
    // Allow server to send message
    printf("Enter message: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    ssize_t sent_len = send(client_socket, buffer, strlen(buffer), 0);
    if (sent_len == -1) {
        perror("Error sending data");
        break;
    }
}
// Close sockets
close(client_socket);
close(server_socket);
return 0;
}

```

(2) Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define SERVER_IP "192.168.111.100"
#define PORT 12345
#define BUFFER_SIZE 1024
int main() {
    int client_socket;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    // Create socket
    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket == -1) {
        perror("Error creating socket");
        exit(EXIT_FAILURE);
    }
    // Set server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);
    server_addr.sin_port = htons(PORT);
    // Connect to server
    if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_addr))
    == -1) {
        perror("Error connecting to server");
        exit(EXIT_FAILURE);
    }
    printf("Connected to server\n");
    while (1) {
        // Send message to server
        printf("Enter message: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        ssize_t sent_len = send(client_socket, buffer, strlen(buffer), 0);
        if (sent_len == -1) {
            perror("Error sending data");
            break;
        }
        // Receive message from server
        ssize_t recv_len = recv(client_socket, buffer, BUFFER_SIZE, 0);
```

```
    if (recv_len <= 0) {
        perror("Error receiving data");
        break;
    }
    buffer[recv_len] = '\0'; // Null-terminate the received data
    printf("Server echoed: %s\n", buffer);
}
// Close socket
close(client_socket);
return 0;
}
```