**연결형 서비스 보고서 [1-1]**컴퓨터 네트워크 02분반

2142851 김형준

**소스코드 분석:** 코드 위에 주석으로 분석 결과를 작성하였습니다.

TCPServer.cpp

// Common.h 헤더 파일을 사용

#include "../Common.h"

// 서버 포트 번호: 9000

// 버퍼 크기: 512

#define SERVERPORT 9000

#define BUFSIZE    512

int main(int argc, char \*argv[])

{

    // 소켓 함수의 반환값을 저장하는 변수 선언

    int retval;

    // 소켓 생성

    SOCKET listen\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

    // 유효한 소켓이 아닌 경우 err\_quit() 함수 호출

    // 오류 출력 후 프로그램 종료

    if (listen\_sock == INVALID\_SOCKET) err\_quit("socket()");

    // bind()

    // 구조체(struct) serveraddr 선언

    // IPv4 + 포트

    struct sockaddr\_in serveraddr;

    // serveraddr의 값을 0으로 초기화

    memset(&serveraddr, 0, sizeof(serveraddr));

    // IPv4를 나타내는 AF\_INET 값 설정

    serveraddr.sin\_family = AF\_INET;

    // 수신할 IP 주소 설정

    // INADDR\_ANY: 서버가 시스템에 있는 모든 네트워크 인터페이스를 통해 들어오는 연결을 수신한다.

    serveraddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    // 수신할 포트 설정

    // 포트: SERVERPORT 값 (#define SERVERPORT 9000)

    serveraddr.sin\_port = htons(SERVERPORT);

    // bind(): 서버 소켓에 주소 할당

    // listen\_sock: 서버 소켓의 파일 디스크립터

    retval = bind(listen\_sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));

    if (retval == SOCKET\_ERROR) err\_quit("bind()");

    // listen()

    // 클라이언트의 접속 대기

    retval = listen(listen\_sock, SOMAXCONN);

    if (retval == SOCKET\_ERROR) err\_quit("listen()");

    // 데이터 통신에 사용할 변수

    // 통신할 소켓, 클라이언트 주소, 버퍼 선언

    SOCKET client\_sock;

    struct sockaddr\_in clientaddr;

    socklen\_t addrlen;

    char buf[BUFSIZE + 1];

    while (1) {

        // accept()

        // 클라이언트 연결 요청이 들어올 시 소켓 생성

        // (그 전까지 대기)

        addrlen = sizeof(clientaddr);

        client\_sock = accept(listen\_sock, (struct sockaddr \*)&clientaddr, &addrlen);

        if (client\_sock == INVALID\_SOCKET) {

            err\_display("accept()");

            break;

        }

        // 접속한 클라이언트 정보 출력

        // 클라이언트 접속 시 IP 주소와 포트 출력

        char addr[INET\_ADDRSTRLEN];

        inet\_ntop(AF\_INET, &clientaddr.sin\_addr, addr, sizeof(addr));

        printf("\n[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",

            addr, ntohs(clientaddr.sin\_port));

        // 클라이언트와 데이터 통신

        while (1) {

            // 데이터 받기

            retval = recv(client\_sock, buf, BUFSIZE, 0);

            // 오류 발생 시 break

            if (retval == SOCKET\_ERROR) {

                err\_display("recv()");

                break;

            }

            // 수신 데이터 크기가 0일 시 break

            else if (retval == 0)

                break;

            // 받은 데이터 출력

            buf[retval] = '\0';

            printf("[TCP/%s:%d] %s\n", addr, ntohs(clientaddr.sin\_port), buf);

            // 데이터 보내기

            retval = send(client\_sock, buf, retval, 0);

            if (retval == SOCKET\_ERROR) {

                err\_display("send()");

                break;

            }

        }

        // 소켓 닫기

        // 연결 종료

        close(client\_sock);

        // 종료 시 메시지 출력

        printf("[TCP 서버] 클라이언트 종료: IP 주소=%s, 포트 번호=%d\n",

            addr, ntohs(clientaddr.sin\_port));

    }

    // 소켓 닫기

    close(listen\_sock);

    return 0;

}

TCPClient.cpp

#include "../Common.h"

// 서버의 IP 주소: 127.0.0.1

char \*SERVERIP = (char \*)"127.0.0.1";

#define SERVERPORT 9000

#define BUFSIZE    512

int main(int argc, char \*argv[])

{

    // 소켓 함수 반환값을 저장할 변수 선언

    int retval;

    // 명령행 인수가 있으면 IP 주소로 사용

    if (argc > 1) SERVERIP = argv[1];

    // 소켓 생성

    // socket() 함수 호출 -> TCP 소켓 생성

    SOCKET sock = socket(AF\_INET, SOCK\_STREAM, 0);

    // 오류 발생 시 메시지 출력 후 종료

    if (sock == INVALID\_SOCKET) err\_quit("socket()");

    // connect()

    // serveraddr: 주소 정보 설정

    struct sockaddr\_in serveraddr;

    // 구조체 serveraddr 0으로 초기화

    memset(&serveraddr, 0, sizeof(serveraddr));

    // AF\_INET: IPv4 주소 체계 사용함

    serveraddr.sin\_family = AF\_INET;

    // inet\_pton(): 문자열 형태의 IP 주소를 이진 형태로 변환 후 sin\_addr에 저장

    inet\_pton(AF\_INET, SERVERIP, &serveraddr.sin\_addr);

    // server의 포트 번호 설정

    serveraddr.sin\_port = htons(SERVERPORT);

    // connect(): 서버 연결

    retval = connect(sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));

    // 연결 실패 시 오류 출력 후 종료

    if (retval == SOCKET\_ERROR) err\_quit("connect()");

    // 데이터 통신에 사용할 변수

    char buf[BUFSIZE + 1];

    // 변수 len: 입력 데이터 길이 저장 변수

    int len;

    // 서버와 데이터 통신

    while (1) {

        // 데이터 입력

        printf("\n[보낼 데이터] ");

        // fgets를 통해 데이터 입력 받고, buf에 저장

        // 입력 실패 시 break

        if (fgets(buf, BUFSIZE + 1, stdin) == NULL)

            break;

        // '\n' 문자 제거

        // 입력 데이터 길이 계산 후 len에 저장

        len = (int)strlen(buf);

        // \n 문자가 있는 경우 제거

        if (buf[len - 1] == '\n')

            buf[len - 1] = '\0';

        // 입력이 없다면 break

        if (strlen(buf) == 0)

            break;

        // 데이터 보내기

        // send(): 데이터를 소켓에 작성

        retval = send(sock, buf, (int)strlen(buf), 0);

        if (retval == SOCKET\_ERROR) {

            err\_display("send()");

            break;

        }

        // 데이터 전송한 경우: 전송 바이트 출력

        printf("[TCP 클라이언트] %d바이트를 보냈습니다.\n", retval);

        // 데이터 받기

        // recv(): 데이터를 소켓에서 읽어옴

        retval = recv(sock, buf, retval, MSG\_WAITALL);

        // 수신 실패 or 연결 종료: err\_display() 출력 후 break

        if (retval == SOCKET\_ERROR) {

            err\_display("recv()");

            break;

        }

        else if (retval == 0)

            break;

        // 받은 데이터 출력

        buf[retval] = '\0';

        printf("[TCP 클라이언트] %d바이트를 받았습니다.\n", retval);

        printf("[받은 데이터] %s\n", buf);

    }

    // 소켓 닫기

    close(sock);

    return 0;

}