**TCP/UDP를 이용한  
server-client 채팅 프로그램 구현**

컴퓨터네트워크 02분반

컴퓨터공학과

2142851 김형준

**목차**

1. **TCP-multi-thread server/client**
   1. TCPChatServer.c
   2. TCPChatClient.c
   3. 출력 결과
2. **UDP-multi-thread server/client**
   1. UDPChatServer.c
   2. UDPChatClient.c
   3. 출력 결과
   4. 문제와 해결
3. **TCP-multi-thread server/client**
4. TCPChatServer.c

*#include* "../Common.h"

*#include* <pthread.h>

*#define* BUFSIZE    512

*#define* SERVERPORT 9000

*// char id[11] = "- Server: ";*

*// 클라이언트로부터 메시지를 수신하는 함수*

void \*receive\_func(void \**arg*) {

    int client\_sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 데이터 받기*

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

*if* (retval == SOCKET\_ERROR) {

            err\_display("recv()");

*break*;

        } *else* *if* (retval == 0) {

            printf("[TCP 서버] 클라이언트와의 연결 종료\n");

*break*;

        }

*// 받은 데이터 출력*

        buf[retval] = '\0';

*// printf("[TCP 서버] %d바이트를 받았습니다.\n", retval);// 받은 데이터 출력*

        printf("- Client: %s\n", buf);

    }

    close(client\_sock);

    pthread\_exit(NULL);

}

*// 클라이언트로 메시지를 보내는 함수*

void \*send\_func(void \**arg*) {

    int client\_sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

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

*break*;

        int len = (int)strlen(buf);

*if* (buf[len - 1] == '\n')

            buf[len - 1] = '\0';

*if* (strlen(buf) == 0)

*break*;

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

*if* (retval == SOCKET\_ERROR) {

            err\_display("send()");

*break*;

        }

    }

    close(client\_sock);

    pthread\_exit(NULL);

}

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

*// 소켓 생성*

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

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

    struct sockaddr\_in serveraddr;

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

    serveraddr.sin\_family = AF\_INET;

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

    serveraddr.sin\_port = htons(SERVERPORT);

*if* (bind(listen\_sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr)) == SOCKET\_ERROR)

        err\_quit("bind()");

*if* (listen(listen\_sock, SOMAXCONN) == SOCKET\_ERROR)

        err\_quit("listen()");

    struct sockaddr\_in clientaddr;

    socklen\_t addrlen = sizeof(clientaddr);

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

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

        err\_display("accept()");

    }

    pthread\_t r\_tid, s\_tid;

*// 클라이언트로부터 메시지를 수신하는 스레드 생성*

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &client\_sock) != 0) {

        printf("[TCP 서버] 수신 스레드 생성 실패\n");

        close(client\_sock);

    }

*// 클라이언트로 메시지를 보내는 스레드 생성*

*if* (pthread\_create(&s\_tid, NULL, send\_func, &client\_sock) != 0) {

        printf("[TCP 서버] 전송 스레드 생성 실패\n");

        close(client\_sock);

    }

    pthread\_join(s\_tid, NULL);

*// 소켓 닫기*

    close(listen\_sock);

*return* 0;

}

1. TCPChatClient.c

*#include* "../Common.h"

*#include* <pthread.h>

*#define* SERVERPORT 9000

*#define* BUFSIZE    512

*// 서버로부터 메시지를 수신하는 함수*

void \*receive\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 데이터 받기*

        retval = recv(sock, buf, BUFSIZE, 0);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recv()");

*break*;

        } *else* *if* (retval == 0) {

            printf("- Server: This host left this chatting room.\n");

*break*;

        }

*// 받은 데이터 출력*

        buf[retval] = '\0';

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

        printf("- Server: %s\n", buf);

    }

    close(sock);

    pthread\_exit(NULL);

}

*// 서버로 메시지를 보내는 함수*

void \*send\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

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

*break*;

        int len = (int)strlen(buf);

*if* (buf[len - 1] == '\n')

            buf[len - 1] = '\0';

*if* (strlen(buf) == 0)

*break*;

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

*if* (retval == SOCKET\_ERROR) {

            err\_display("send()");

*break*;

        }

    }

    close(sock);

    pthread\_exit(NULL);

}

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

    int retval;

*// 명령행 인수가 있으면 IP 주소로 사용*

    char \*SERVERIP = (argc > 1) ? argv[1] : (char \*)"127.0.0.1";

*// 소켓 생성*

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

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

*// connect()*

    struct sockaddr\_in serveraddr;

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

    serveraddr.sin\_family = AF\_INET;

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

    serveraddr.sin\_port = htons(SERVERPORT);

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

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

    pthread\_t r\_tid, s\_tid;

*// 서버로부터 메시지를 수신하는 스레드 생성*

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &sock) != 0) {

        printf("[TCP 클라이언트] 수신 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

*// 서버로 메시지를 보내는 스레드 생성*

*if* (pthread\_create(&s\_tid, NULL, send\_func, &sock) != 0) {

        printf("[TCP 클라이언트] 전송 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

    pthread\_join(s\_tid, NULL);

*// 소켓 닫기*

    close(sock);

*return* 0;

}

1. 출력 결과

<TCPChatClient>

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

<TCPChatServer>

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. **UDP-multi-thread server/client**
2. UDPChatServer.c

*#include* "../Common.h"

*// char \*IP = (char \*)"127.0.0.1";*

*#define* SERVERPORT 9000

*#define* BUFSIZE    512

*#include* <pthread.h>

pthread\_mutex\_t clientaddr\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

struct sockaddr\_in clientaddr; *// 전역 변수로 선언*

void \*receive\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    socklen\_t addrlen = sizeof(clientaddr);

    int retval;

*while* (1) {

*// recvfrom() 호출*

        retval = recvfrom(sock, buf, BUFSIZE, 0, (struct sockaddr \*)&clientaddr, &addrlen);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recvfrom()");

*break*;

        }

*// 받은 데이터 출력*

        buf[retval] = '\0';

        printf("- client[%s:%d]: %s\n", inet\_ntoa(clientaddr.sin\_addr), ntohs(clientaddr.sin\_port), buf);

    }

    pthread\_exit(NULL);

}

void \*send\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 사용자로부터 데이터 입력 받기*

*// printf("\n[보낼 데이터] ");*

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

*break*;

        int len = (int)strlen(buf);

*if* (buf[len - 1] == '\n')

            buf[len - 1] = '\0';

*if* (strlen(buf) == 0)

*break*;

*// 동기화: 전역 변수 clientaddr 사용 시 동기화 처리*

        pthread\_mutex\_lock(&clientaddr\_mutex);

        retval = sendto(sock, buf, len, 0, (struct sockaddr \*)&clientaddr, sizeof(clientaddr));

        pthread\_mutex\_unlock(&clientaddr\_mutex);

*if* (retval == SOCKET\_ERROR) {

            err\_display("sendto()");

*break*;

        }

*// printf("[UDP 서버] %d바이트를 보냈습니다.\n", retval);*

    }

    pthread\_exit(NULL);

}

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

{

*// 소켓 생성*

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

*if* (sock == INVALID\_SOCKET)

        err\_quit("socket()");

*// bind()*

    struct sockaddr\_in serveraddr;

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

    serveraddr.sin\_family = AF\_INET;

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

    serveraddr.sin\_port = htons(SERVERPORT);

*// 바인딩*

*if* (bind(sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr)) == SOCKET\_ERROR)

        err\_quit("bind()");

*// 수신 및 전송 스레드 생성*

    pthread\_t r\_tid, s\_tid;

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &sock) != 0) {

        printf("[UDP 서버] 수신 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

*if* (pthread\_create(&s\_tid, NULL, send\_func, &sock) != 0) {

        printf("[UDP 서버] 전송 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

    pthread\_join(s\_tid, NULL);

    close(sock);

*return* 0;

}

1. UDPChatClient.c

*#include* "../Common.h"

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

*#define* SERVERPORT 9000

*#define* BUFSIZE    512

*// 수신 스레드 함수*

void \*receive\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    struct sockaddr\_in serveraddr;

    socklen\_t addrlen = sizeof(serveraddr);

    int retval;

*while* (1) {

        retval = recvfrom(sock, buf, BUFSIZE, 0, (struct sockaddr \*)&serveraddr, &addrlen);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recvfrom()");

*break*;

        }

        buf[retval] = '\0';

        printf("- server[%s:%d]: %s\n", inet\_ntoa(serveraddr.sin\_addr), ntohs(serveraddr.sin\_port), buf);

    }

    pthread\_exit(NULL);

}

*// 전송 스레드 함수*

void \*send\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

    struct sockaddr\_in serveraddr;

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

    serveraddr.sin\_family = AF\_INET;

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

    serveraddr.sin\_port = htons(SERVERPORT);

*while* (1) {

*// printf("\n[보낼 데이터] ");*

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

*break*;

        int len = (int)strlen(buf);

*if* (buf[len - 1] == '\n')

            buf[len - 1] = '\0';

*if* (strlen(buf) == 0)

*break*;

        retval = sendto(sock, buf, len, 0, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));

*if* (retval == SOCKET\_ERROR) {

            err\_display("sendto()");

*break*;

        }

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

    }

    pthread\_exit(NULL);

}

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

{

*if* (argc > 1) {

        SERVERIP = argv[1];

    }

*// UDP 소켓 생성*

    int sock = socket(AF\_INET, SOCK\_DGRAM, 0);

*if* (sock == INVALID\_SOCKET) {

        err\_quit("socket");

    }

*// 수신 및 전송 스레드 생성*

    pthread\_t r\_tid, s\_tid;

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &sock) != 0) {

        printf("[UDP 클라이언트] 수신 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

*if* (pthread\_create(&s\_tid, NULL, send\_func, &sock) != 0) {

        printf("[UDP 클라이언트] 전송 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

    pthread\_join(s\_tid, NULL);

    close(sock);

*return* 0;

}

1. 출력 결과

<UDPChatClient>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<UDPChatServer>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 문제와 해결

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Server 쪽에서 데이터를 입력하여 보냈을 때,

sendto() 함수의 에러가 났습니다.

printf("sendto() 호출 전:\n");

printf("버퍼 내용: %s\n", buf);

printf("버퍼 길이: %d\n", len);

printf("소켓 파일 디스크립터: %d\n", sock);

printf("클라이언트 주소 정보: %s:%d\n", inet\_ntoa(clientaddr.sin\_addr), ntohs(clientaddr.sin\_port));

printf("주소 구조체 크기: %d\n", addrlen);

따라서 위 코드를 추가하여 sendto() 함수 안의 값을 확인하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

여기서 클라이언트 주소 정보가 이상하게 되어 있음을 알 수 있었습니다.

따라서 주소 정보를 전역 변수로 설정하고, 초기화 부분을 수정하여 고쳤습니다.