데이터 통신
Project1 : UDP echo

컴퓨터 전공
김다빈 2015004375

Project1 : UDP echo

1. 결과 화면

```
➜ data_C ./echo_server 1245
client wait....
Recv from 'Client' : hello
Send to 'Client' : hello
Recv from 'Client' : yeah
Send to 'Client' : yeah
Recv from 'Client' : i did it you know
Send to 'Client' : i did it you know
Recv from 'Client' : i can do it i think it's so nice
Send to 'Client' : i can do it i think it's so nice
^Z
[6]  + 83347 suspended  ./echo_server 1245
```

```
1 warning generated.
➜ data_C ./tmp_client 127.0.0.1 1245
Please type the TEXT : hello
[Echo] hello
Please type the TEXT : yeah
[Echo] yeah
Please type the TEXT : i did it you know
[Echo] i did it you know
Please type the TEXT : i can do it i think it's so nice
[Echo] i can do it i think it's so nice
Please type the TEXT : ^Z
[20]  + 83397 suspended  ./tmp_client 127.0.0.1 1245
```

- server 실행시 “client wait…” 실행 : client가 들어올 때 까지 기다린다.
- client에서 들어오고 입력을 실행하면 “Please type the TEXT”가 뜨면 에코할 TEXT를 입력한다.
- server에서 Recv함을 확인하고 Send함까지 확인했다는 메세지를 입력한다.
- Server에서 다시 에코해서 보낸 메세지를 client에서 [Echo]뒤에 메세지를 띄운다.
- +) 포트번호, ip 직접 입력 ex : ./echo_server 포트번호

2. 코드 설명

- Server Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>
#include <pthread.h>

#define MAXLINE 1024 //buf 크기

void *thread_func(void *arg); //쓰레드 시작 함수
int listen_sock, accp_sock;
struct sockaddr_in server_addr, client_addr;

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

    int addrlen = sizeof(server_addr);
    int i, status ;
    pthread_t thr_id;
    pid_t pid;

    if(argc != 2) {
        printf("Use %s PortNumber\n", argv[0]);
        exit(0);
    }

    //create socket
    if((listen_sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket Fail");
        exit(0);
    }

    //server
    memset(&server_addr, 0, sizeof(server_addr)); //0으로 초기화
    memset(&client_addr,0,sizeof(client_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(atoi(argv[1]));

    //bind 호출
    if(bind(listen_sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("bind Fail");
        exit(0);
    }


    listen(listen_sock, 10);

    puts("client wait....");

    accp_sock = accept(listen_sock, (struct sockaddr *)&client_addr, &addrlen);
    //printf("1\n");
    if(accp_sock < 0) {
        perror("accept fail");
        exit(0);
    }
```

1.서버를 먼저 실행하기 위하여 create socket부분에 소켓을 생성하고, 제대로 생성이 안됐을 때를 체크해준다.
2. server_address, client_address 등 기본 서버 세팅을 해준다.(//server 부분)
3. 생성 소켓에 로컬 주소를 할당 성공여부를 반환한다.(//bind)
4. 해당 소켓의 연결을 기다린다. 연결 성공시 0을 반환한다.(//listen)
5. client가 연결 요청이 들어 왔을 시, 연결을 받아 들인다.(//accept)
6. 연결이 이루어진 후 두개의 스레드는 각각 thread_recv와 thread_send를 통해서 client에서 주소를 받아오고 다시 되돌려주게 된다.
7. 첫번째 스레드는(thr_id[0]) thread_recv함수를 통해

```
    while(1){

       // pthread_mutex_lock(&mutex);
        if((status = pthread_create(&thr_id[0], NULL,&thread_recv, &accp_sock))!= 0) {
            printf("#1 Thread create error: %s\n", strerror(status));
            exit(0);
        }
        //pthread_mutex_unlock(&mutex);
        //printf("2\n");


    //인자로 지정한 스레드 id가 종료하기를 기다립니다.
        //printf("3\n");
        pthread_join(thr_id[0], NULL);

        if((status = pthread_create(&thr_id[1], NULL,&thread_send, &accp_sock))!= 0) {
            printf("#2 Thread create error: %s\n", strerror(status));
            exit(0);
        }

        pthread_join(thr_id[1], NULL);


    }

    close(accp_sock);
    return 0;
}

char buf[MAXLINE+1];
int nbyte;

void *thread_recv(void *arg) {

    int accp_sock=(int) *((int*) arg);
    int addrlen;
    int status;

    addrlen=sizeof(client_addr);


    nbyte=recvfrom(accp_sock,buf,MAXLINE,0,(struct sockaddr*)&client_addr,&addrlen);
    if(nbyte<0){
        perror("Recv Fail");
        exit(0);
    }

    buf[nbyte]='\0';
    printf("Recv from 'Client' : %s\n",buf);

    //pthread_mutex_lock(&mutex);

    //pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
    close(accp_sock);

}
void *thread_send(void *arg) {

    int accp_sock=(int) *((int*) arg);
    int addrlen;

    addrlen=sizeof(client_addr);

    nbyte=sendto(accp_sock,buf,nbyte,0,(struct sockaddr*)&client_addr,sizeof(client_addr));
    if(nbyte<0){
        perror("Send Fail");
        exit(0);
    }
    printf("Send to 'Client' : %s\n",buf);

    memset(buf,0,MAXLINE+1);

    pthread_exit(NULL);
    close(accp_sock);

}
```

recvfrom을 이용하여 client에서 text를 buf에 받아 저장하게 된다. 그리고 스레드가 프로세스를 끝내면 스레드를 exit 하게 된다.

8. 두번째 스레드(thr_id[1])에서는 thread_send에서 sendto를 통해 client에 buf에 있던 받아온 내용을 보내주고 스레드를 종료한다.

- Client Code

```c
#include <sys/types.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>

struct sockaddr_in server_addr;

#define BUFSIZE 1024

void *thread_func(void *arg);

int main(int argc, char **argv)
{
    int sock;
    char message[BUFSIZE];
    int message_len, recv_len, recv_num;

    pthread_t thr_id;
    pid_t pid;
    int status;

    if (argc != 3)
    {
        printf("usage: %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    /* Create Socket */
    sock = socket(PF_INET, SOCK_STREAM, 0);
    if (sock == -1)
        exit(0);

    /* Address Setting */
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(argv[1]);
    server_addr.sin_port = htons(atoi(argv[2]));

    /* Connect to Server */
    if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr))
        == -1)
        exit(-1);
```

```c
    while(1){

        if((status = pthread_create(&thr_id[0], NULL,&thread_send, &sock))!= 0) {
            printf("#1 Thread create error: %s\n", strerror(status));
            exit(0);
        }

        //인자로 지정한 스레드 id가 종료하기를 기다립니다.
        pthread_join(thr_id[0], NULL);

        if((status = pthread_create(&thr_id[1], NULL,&thread_recv, &sock))!= 0) {
            printf("#2 Thread create error: %s\n", strerror(status));
            exit(0);
        }

        pthread_join(thr_id[1], NULL);


    }

    close(sock);

    return 0;
}

char buf[BUFSIZE+1];
int nbyte;

void *thread_send(void *arg) {

    int accp_sock=(int) *((int*) arg);
    int len;
    int status;

    printf("Please type the TEXT : ");
    if(fgets(buf,BUFSIZE+1,stdin)==NULL) exit(0);
    len=strlen(buf);

    if(buf[len-1]=='\n') buf[len-1]='\0';
    if(strlen(buf)==0){
        printf("No TEXT to send.\n");
        exit(0);
    }

    nbyte=sendto(accp_sock,buf,strlen(buf),0,(struct sockaddr*)&server_addr,sizeof(server_addr));
    if(nbyte<0){
        perror("Send Fail");
        exit(0);
    }

    pthread_exit(NULL);
    close(accp_sock);

}
```

1. 통신을 위한 소켓을 생성한다.
2. 생성 후 기본 서버 세팅을 해준다.
3. connect를 통해서 서버 프로그램과 연결해 준다.
4. 서버와 동일하게 두개의 송수신 스레드를 생성. thread_recv와 thread_send를 통해 데이터를 보대고 받게 된다.
5. 첫번째 스레드(thr_id[0])는 thread_send 함수에서 에코될 데이터를 입력하고 데이터를 sendto 함수를 통해서 서버에 전송하게 된다.
6. 두번째 스레드(thr_id[1])는 thread_recv 함수에서 서버가 되돌려준 데이터를 recvfrom으로 받아 다시 client창에 표시해준다.
7. 프로그램 종료 전까지 서버와 클라이언트 에코를 반복한다.

```c
void *thread_recv(void *arg) {

    int accp_sock=(int) *((int*) arg);
    int addrlen;

    addrlen=sizeof(server_addr);

    nbyte=recvfrom(accp_sock,buf,BUFSIZE,0,(struct sockaddr*)&server_addr,&addrlen);
    if(nbyte<0){
        perror("Recv Fail");
        exit(0);
    }

    buf[nbyte]='\0';
    printf("[Echo] %s\n",buf);

    memset(buf,0,BUFSIZE+1);

    pthread_exit(NULL);
    close(accp_sock);

}
```