컴퓨터네트워크및실습 TermProject

21720903 조태식

Chat\_serv.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <pthread.h>

#define BUF\_SIZE 100 //글자수 100제한

#define MAX\_CLNT 256 //접속자수 256제한

void \* handle\_clnt(void \* arg); //미리선언

void send\_msg(char \* msg, int len); //미리선언

void error\_handling(char\* msg); //미리선언

int clnt\_cnt=0; //접속한사람 몇명인지 변수

int clnt\_socks[MAX\_CLNT]; //생성된 소켓의 파일디스크립터 저장 배열

pthread\_mutex\_t mutx; //스레드를 위한 뮤텍스 선언

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

{

int serv\_sock, clnt\_sock; //선언

struct sockaddr\_in serv\_adr, clnt\_adr; //주소,포트,ip갖춘 구조체 선언

int clnt\_adr\_sz; //클라이언트 주소 사이즈 선언

pthread\_t t\_id; //쓰레드 선언

if(argc!=2) { //만약 ./c 포트번호가 아닐경우

printf("Usage : %s <port>\n", argv[0]); //이렇게 입력해주세요!!

exit(1); //종료

}

pthread\_mutex\_init(&mutx, NULL); //뮤텍스 초기화

serv\_sock=socket(PF\_INET, SOCK\_STREAM, 0); //ipv4프로토콜 사용,tcp프로토콜사용,0을 매개변수로 한 socket()함수에 대한 디스크립러 반환

//-1이 반환되면 소켓 생성실패.

memset(&serv\_adr, 0, sizeof(serv\_adr)); //&serv\_adr부터 serv\_adr의 크기만큼 0으로 세팅

serv\_adr.sin\_family=AF\_INET; //주소체계에 AF\_INET저장

serv\_adr.sin\_addr.s\_addr=htonl(INADDR\_ANY); //sin\_addr중 구조체인 s\_adrr에 저장

//4byte데이터를 네트워크 byte order로 변경.

//이 컴퓨터에서 존재하는 랜카드중 사용가능한 랜카드ip주소 사용.

serv\_adr.sin\_port=htons(atoi(argv[1])); //argv[1]로 받은 문자열을 숫자로 변환해서 port에 저장

//2byte데이터를 네트워크 byte order로 변경.

if(bind(serv\_sock, (struct sockaddr\*) &serv\_adr, sizeof(serv\_adr))==-1) //서버소켓과 &serv)adr구조체, 주소구조체의 길이로 bind()함수 호출 성공하면 0 실패하면 -1

error\_handling("bind() error"); //bind()에러

if(listen(serv\_sock, 5)==-1) //serv\_sock과 연결요구 개수의 최대값 5로 지정 성공시 0반화, 실패시 -1 반환, 가볍게 말하면 놀이공원에 한번에 줄을 설 수 있는 인원이다

//256으로도 변경가능

error\_handling("listen() error"); //listen()에러

while(1) //계속 반복

{

clnt\_adr\_sz=sizeof(clnt\_adr); //clnt\_adr의 크기를 clnt\_adr\_sz에 저장

clnt\_sock=accept(serv\_sock, (struct sockaddr\*)&clnt\_adr,&clnt\_adr\_sz); //serv\_sock에 clnt가 연결이 되는지 확인, 실패시 -1 반환

pthread\_mutex\_lock(&mutx); //임계영역에 진입하기 위해 잠금

clnt\_socks[clnt\_cnt++]=clnt\_sock; //accpet()받은 clnt\_sock를 clont\_socks배열에 저장

pthread\_mutex\_unlock(&mutx); //잠금 해제

pthread\_create(&t\_id, NULL, handle\_clnt, (void\*)&clnt\_sock); //쓰레드 생성, 기본특성위해 NULL, 쓰레드 함수는 handle\_clnt, 그리고 accpet()에서 받은 &clnt\_sock

pthread\_detach(t\_id); //쓰레드 종료될때 모든 자원 해제

printf("Connected client IP: %s \n", inet\_ntoa(clnt\_adr.sin\_addr)); //inet\_nota를 이용하여 네트워크 바이트 순서의 32비트값을 변환

}

close(serv\_sock); //서버소켓 종료

return 0; //끝

}

void \* handle\_clnt(void \* arg) //핸들함수 매개변수 \*arg

{

int clnt\_sock=\*((int\*)arg); //위의 pthread\_create에서 (void\*)&clnt\_sock으로 매개변수를 주었으므로 int clnt\_sock에 넣기위해서 int형 변환

int str\_len=0, i; //문자열 길이, 반복할때 사용할 i선언

char msg[BUF\_SIZE]; //BUF\_SIZE만큼의 메시지문자배열 선언

while((str\_len=read(clnt\_sock, msg, sizeof(msg)))!=0) //클라이언트에서 보낸 메시지를 메시지 크기만큼 읽음, 0이 될때까지(즉 Q나 q를 입력하면)

send\_msg(msg, str\_len); //send\_msg함수 호출

pthread\_mutex\_lock(&mutx); //임계영역에 진입하기 위해 잠금

for(i=0; i<clnt\_cnt; i++) // remove disconnected client

{

if(clnt\_sock==clnt\_socks[i]) //q로 소켓종료한 clnt\_socks에서 찾기

{

while(i <clnt\_cnt-1) //0 1 2 3 4 에서 1이 나가면 0 2 3 4 가 되므로 뒤에 있는 소켓들 앞으로 옮기기 하나가 빠졌으므로

{

clnt\_socks[i]=clnt\_socks[i+1]; //옮기기

i++; //다음 소켓

}

break; //for문 종료

}

}

clnt\_cnt--; //하나가 종료되었으므로 -1

pthread\_mutex\_unlock(&mutx); //잠금해제

close(clnt\_sock); //사라진 소켓 종료

return NULL; //NULL반환

}

void send\_msg(char \* msg, int len) // send to all

{

int i; //반복 변수

pthread\_mutex\_lock(&mutx); //임계영역에 진입하기 위해 잠금

for(i=0; i<clnt\_cnt; i++) //반복시작

write(clnt\_socks[i], msg, len); //모든 클라이언트에게 msg보내기

pthread\_mutex\_unlock(&mutx); //잠금 해제

}

void error\_handling(char \* msg) //에러메시지 출력함수

{

fputs(msg, stderr); //msg입력

fputc('\n', stderr); //개행

exit(1); //종료

}

Chat\_clnt.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <pthread.h>

#define BUF\_SIZE 100 //msg사이즈 100제한

#define NAME\_SIZE 20 //이름사이즈 20제한

void \* send\_msg(void \* arg); //미리선언

void \* recv\_msg(void \* arg); //미리선언

void error\_handling(char \* msg); //미리선언

char name[NAME\_SIZE]="[DEFAULT]"; //name문자배열에 디폴드값 저장

char msg[BUF\_SIZE]; //100크기의 문자msg배열선언

int main(int argc, char \*argv[]) //매개변수 받는 main함수

{

int sock; //sock변수 선언

struct sockaddr\_in serv\_addr; //주소,포트,ip갖춘 구조체 선언

pthread\_t snd\_thread, rcv\_thread; //받는 쓰레드와 보내는 쓰레드 선언

void \* thread\_return; //pthread\_join에 사용

if(argc!=4) { //인자가 4개가 아니라면

printf("Usage : %s <IP> <port> <name>\n", argv[0]); //ex)./chat\_clnt 127.0.0.1 9999 one 입력해주세요!

exit(1); //종료

}

sprintf(name, "[%s]", argv[3]); //name에 매개변수3번째것이 문자열로 저장

sock=socket(PF\_INET, SOCK\_STREAM, 0); //ipv4프로토콜 사용,tcp프로토콜사용,0을 매개변수로 한 socket()함수에 대한 디스크립러 반환

//-1이 반환되면 소켓 생성실패.

memset(&serv\_addr, 0, sizeof(serv\_addr)); //&serv\_adr부터 serv\_adr의 크기만큼 0으로 세팅

serv\_addr.sin\_family=AF\_INET; ////주소체계에 AF\_INET저장

serv\_addr.sin\_addr.s\_addr=inet\_addr(argv[1]); //argv[1]로 받은 문자열을 빅엔디안32bit로 변환해서 sin\_addr.s\_addr에 저장

serv\_addr.sin\_port=htons(atoi(argv[2])); //argv[2]로 받은 문자열을 숫자로 변환해서 port에 저장

//2byte데이터를 네트워크 byte order로 변경.

if(connect(sock, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr))==-1) //서버로 접속 요청 실패시 -1 반환

error\_handling("connect() error"); //에러

pthread\_create(&snd\_thread, NULL, send\_msg, (void\*)&sock); //쓰레드 생성, 기본특성위해 NULL, 쓰레드 함수는 send\_msg, 그리고 socket()에서 받은 &sock

pthread\_create(&rcv\_thread, NULL, recv\_msg, (void\*)&sock); //쓰레드 생성, 기본특성위해 NULL, 쓰레드 함수는 recv\_msg, 그리고 socket()에서 받은 &sock

pthread\_join(snd\_thread, &thread\_return); //snd\_thread종료 대기

pthread\_join(rcv\_thread, &thread\_return); //rcv\_thraed종료 대기

close(sock); //소켓종료

return 0; //리턴

}

void \* send\_msg(void \* arg) // send thread main

{

int sock=\*((int\*)arg); //위의 pthread\_create에서 (void\*)&sock 으로 매개변수를 주었으므로 int sock에 넣기위해서 int형 변환

char name\_msg[NAME\_SIZE+BUF\_SIZE]; //NAME\_SIZE와BUF\_SIZE를 합한것크기의 name\_msg문자배열선언

while(1) //반복

{

fgets(msg, BUF\_SIZE, stdin); //입력받은걸 BUF\_SIZE만큼 msg저장

if(!strcmp(msg,"q\n")||!strcmp(msg,"Q\n")) //만약 q나 Q가 입력시

{

close(sock); //소켓종료

exit(0); //종료

}

sprintf(name\_msg,"%s %s", name, msg); //name\_msg에 name과 이어서 msg 저장

write(sock, name\_msg, strlen(name\_msg)); //서버로 메시지 보내기

}

return NULL; //return NULL

}

void \* recv\_msg(void \* arg) // read thread main

{

int sock=\*((int\*)arg); //위의 pthread\_create에서 (void\*)&sock 으로 매개변수를 주었으므로 int sock에 넣기위해서 int형 변환

char name\_msg[NAME\_SIZE+BUF\_SIZE]; //NAME\_SIZE와BUF\_SIZE를 합한것크기의 name\_msg문자배열선언

int str\_len; //문자열길이 선언

while(1) //반복

{

str\_len=read(sock, name\_msg, NAME\_SIZE+BUF\_SIZE-1); //서버에서 보낸 name\_msg를 NAME\_SIZE+BUF\_SIZE-1만큼 읽음

//실패시 -1 반환

if(str\_len==-1) //read실패시

return (void\*)-1;

name\_msg[str\_len]=0; //마지막 값에 NULL저장 종료알리기위해서

fputs(name\_msg, stdout); //name\_msg출력

}

return NULL; //return nULL

}

void error\_handling(char \*msg) //에러메시지 출력함수

{

fputs(msg, stderr); //msg입력

fputc('\n', stderr); //개행

exit(1); //종료

}