[ 게임 네트워크 프로그래밍 ]

**데이터 통신을 구현한 프로그램 소스 코드 작성**

인하대학교 미래인재개발원 문화콘텐츠프로그래밍

김예슬

**TCP / IP 통신을 구현한 프로그램 소스 코드**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#include <process.h>

#define BUF\_SIZE 2048

#define BUF\_SMALL 100

unsigned WINAPI RequestHandler(void\* arg);

char\* ContentType(char\* file);

void SendData(SOCKET sock, char\* ct, char\* fileName);

void SendErrorMSG(SOCKET sock);

void ErrorHandling(char \*message);

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

{

WSADATA wsaData;

SOCKET hServSock, hClntSock;

SOCKADDR\_IN servAdr, clntAdr;

HANDLE hThread;

DWORD dwThreadID;

int clntAdrSize;

if(argc!=2) {

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

exit(1);

}

if(WSAStartup(MAKEWORD(2, 2), &wsaData)!=0)

ErrorHandling("WSAStartup() error!");

hServSock=socket(PF\_INET, SOCK\_STREAM, 0);

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

servAdr.sin\_family=AF\_INET;

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

servAdr.sin\_port=htons(atoi(argv[1]));

if(bind(hServSock, (SOCKADDR\*) &servAdr, sizeof(servAdr))==SOCKET\_ERROR)

ErrorHandling("bind() error");

if(listen(hServSock, 5)==SOCKET\_ERROR)

ErrorHandling("listen() error");

while(1)

{

clntAdrSize=sizeof(clntAdr);

hClntSock=accept(hServSock, (SOCKADDR\*)&clntAdr, &clntAdrSize);

printf("Connection Request : %s:%d\n",

inet\_ntoa(clntAdr.sin\_addr), ntohs(clntAdr.sin\_port));

hThread=(HANDLE)\_beginthreadex(

NULL, 0, RequestHandler, (void\*)hClntSock, 0, (unsigned \*)&dwThreadID);

}

closesocket(hServSock);

WSACleanup();

return 0;

}

unsigned WINAPI RequestHandler(void \*arg)

{

SOCKET hClntSock=(SOCKET)arg;

char buf[BUF\_SIZE];

char method[BUF\_SMALL];

char ct[BUF\_SMALL];

char fileName[BUF\_SMALL];

recv(hClntSock, buf, BUF\_SIZE, 0);

if(strstr(buf, "HTTP/")==NULL)

{

SendErrorMSG(hClntSock);

closesocket(hClntSock);

return 1;

}

strcpy(method, strtok(buf, " /"));

if(strcmp(method, "GET"))

SendErrorMSG(hClntSock);

strcpy(fileName, strtok(NULL, " /"));

strcpy(ct, ContentType(fileName));

SendData(hClntSock, ct, fileName);

return 0;

}

void SendData(SOCKET sock, char\* ct, char\* fileName)

{

char protocol[]="HTTP/1.0 200 OK\r\n";

char servName[]="Server:simple web server\r\n";

char cntLen[]="Content-length:2048\r\n";

char cntType[BUF\_SMALL];

char buf[BUF\_SIZE];

FILE\* sendFile;

sprintf(cntType, "Content-type:%s\r\n\r\n", ct);

if((sendFile=fopen(fileName, "r"))==NULL)

{

SendErrorMSG(sock);

return;

}

send(sock, protocol, strlen(protocol), 0);

send(sock, servName, strlen(servName), 0);

send(sock, cntLen, strlen(cntLen), 0);

send(sock, cntType, strlen(cntType), 0);

while(fgets(buf, BUF\_SIZE, sendFile)!=NULL)

send(sock, buf, strlen(buf), 0);

closesocket(sock);

}

void SendErrorMSG(SOCKET sock)

{

char protocol[]="HTTP/1.0 400 Bad Request\r\n";

char servName[]="Server:simple web server\r\n";

char cntLen[]="Content-length:2048\r\n";

char cntType[]="Content-type:text/html\r\n\r\n";

char content[]="<html><head><title>NETWORK</title></head>"

"<body><font size=+5><br>오류 발생! 요청 파일명 및 요청 방식 확인!"

"</font></body></html>";

send(sock, protocol, strlen(protocol), 0);

send(sock, servName, strlen(servName), 0);

send(sock, cntLen, strlen(cntLen), 0);

send(sock, cntType, strlen(cntType), 0);

send(sock, content, strlen(content), 0);

closesocket(sock);

}

char\* ContentType(char\* file)

{

char extension[BUF\_SMALL];

char fileName[BUF\_SMALL];

strcpy(fileName, file);

strtok(fileName, ".");

strcpy(extension, strtok(NULL, "."));

if(!strcmp(extension, "html")||!strcmp(extension, "htm"))

return "text/html";

else

return "text/plain";

}

void ErrorHandling(char\* message)

{

fputs(message, stderr);

fputc('\n', stderr);

exit(1);

}

**UDP 통신을 구현한 프로그램 소스 코드**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#define MAXLINE 511

#define BLOCK 255

#define FILENAME "buf.txt"

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

struct sockaddr\_in servaddr, cliaddr;

int s, nbyte, addrlen = sizeof(struct sockaddr);

char buf[MAXLINE+1];

FILE \*stream;

if(argc != 2) {

printf("usage: %s port\n", argv[0]);

exit(0);

}

if((s = socket(PF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("socket fail");

exit(0);

}

memset(&cliaddr, 0, addrlen); //bzero((char \*)&cliaddr, addrlen);

memset(&servaddr, 0, addrlen); //bzero((char \*)&servaddr,addrlen);

servaddr.sin\_family = AF\_INET;

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

servaddr.sin\_port = htons(atoi(argv[1]));

if(bind(s, (struct sockaddr \*)&servaddr, addrlen) < 0) {

perror("bind fail");

exit(0);

}

if((stream = fopen(FILENAME, "w")) == 0) {

printf("Faile open error\n");

exit(1);

}

while(1)

{

puts("Server : waiting request.");

nbyte = recvfrom(s, buf, MAXLINE , 0, (struct sockaddr \*)&cliaddr, &addrlen);

if(nbyte< 0) {

perror("recvfrom fail");

exit(1);

}

buf[nbyte] = 0;

if(!strncmp(buf, "end of file", 10)) {

printf("file close");

fclose(stream);

break;

} else {

printf("%d byte recv: %s\n",nbyte, buf);

fputs(buf, stream);

}

puts("sendto complete");

}

if((stream = fopen(FILENAME, "r")) == NULL) {

printf("Read File Error");

exit(1);

}

while(!feof(stream)) {

buf[0] = '\0';

fgets(buf, BLOCK, stream);

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

if(sendto(s, buf, strlen(buf), 0, (struct sockaddr \*)&cliaddr, addrlen) < 0) {

perror("sendto fail");

exit(0);

}

}

fclose(stream);

close(s);

return 0;

}