|  |
| --- |
| **일일 업무 사항 정리** |

|  |  |
| --- | --- |
| **작성자** | 제품팀 이민성 인턴 |
| **업무 일시** | 20230130~20230224 |

|  |
| --- |
| **세부 사항** |
| **1. 업무 내역 요약 정리**   |  |  | | --- | --- | | **목표 내역** | **Done & Plan** | | **1. 틱택토 게임을 두 사용자가 한 컴퓨터에서 플레이 할 수 있게 하기**  **2. 네트워크 프로그래밍을 이용해 틱택토 게임을 두 사용자가 서로 다른 컴퓨터에서 플레이 할 수 있게 하기** | **1. tictactoe.c 코드 정리본**  **2. 틱택토 아키텍처**  **3. 틱택토 게임을 두 사용자가 한 컴퓨터에서 플레이 할 수 있게 하기**  **4. server.c, client.c 분석 (간단한 문자열 전송 코드)**  **4-1. atoi**  **4-2. memset**  **4-3. sockaddr, sockaddr\_in**  **4-4. write**  **4-5. hton, ntoh**  **4-6. INADDR\_ANY**  **4-7. AF\_INET, PF\_INET**  **4-8. SOCK\_STREAM**  **4-9. 주소 구조체 종류**  **4-10. fputs, fputc**  **4-11. sizeof**  **4-12. argc, argv**  **5.** **네트워크 프로그래밍을 이용해 틱택토 게임을 두 사용자가 서로 다른 컴퓨터에서 플레이 할 수 있게 하기**  **5-1. 실패 코드**  **5-2. 실패 원인이 무엇일까?** | |

**2. 내용 세부 (업무 세부 내역 정리 및 기타 사항 정리)**

**1. tictactoe.c 코드 정리본**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int board**[**3**][**3**]** **=** **{**0**,};**

int i**,** j**,** flag**,** win**,** xin**,** yin**,** x**,** y**;**

int PrintBoard**(**void**)** // 보드판 출력 함수

**{**

system**(**"clear"**);**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 1**)**

printf**(**"O |"**);**

**else** **if(**board**[**i**][**j**]** **==** 2**)**

printf**(**"X |"**);**

**else** **if(**board**[**i**][**j**]** **==** 0**)**

printf**(**" |"**);**

**}**

printf**(**"\n"**);**

**}** // 출력 부분

**return** 0**;**

**}**

int Check**(**void**)** // 검사 함수

**{**

flag **=** 0**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 가로

**{**

**if((**board**[**i**][**0**]** **==** board**[**i**][**1**])** **&&** **(**board**[**i**][**1**]** **==** board**[**i**][**2**]))**

**{**

**if(**board**[**i**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**i**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 세로

**{**

**if((**board**[**0**][**i**]** **==** board**[**1**][**i**])** **&&** **(**board**[**1**][**i**]** **==** board**[**2**][**i**]))**

**{**

**if(**board**[**1**][**i**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**i**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**0**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**2**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**2**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**0**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**return** 0**;**

**}**

int Ai**(**void**)** // 인공지능: 2개 이상 이어진 것을 찾아서 검사해서 한쪽에 놓는다.

**{**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 반복

**{**

**if((**board**[**0**][**0**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** 1**)** **&&** **(**board**[**2**][**2**]** **==** 0**))** // 대각선

**{**board**[**2**][**2**]** **=** 2**;** **break;}**

**else** **if((**board**[**1**][**1**]** **==** board**[**2**][**2**])** **&&** **(**board**[**1**][**1**]** **==** 1**)** **&&** **(**board**[**0**][**0**]** **==** 0**))** // 대각선

**{**board**[**0**][**0**]** **=** 2**;** **break;}**

**else** **if((**board**[**0**][**2**]** **==** board**[**1**][**1**])** **&&** **(**board**[**0**][**2**]** **==** 1**)** **&&** **(**board**[**2**][**0**]** **==** 0**))** // 대각선

**{**board**[**2**][**0**]** **=** 2**;** **break;}**

**else** **if((**board**[**2**][**0**]** **==** board**[**1**][**1**])** **&&** **(**board**[**2**][**0**]** **==** 1**)** **&&** **(**board**[**0**][**2**]** **==** 0**))** // 대각선

**{**board**[**0**][**2**]** **=** 2**;** **break;}**

**else** **if((**board**[**0**][**0**]** **==** board**[**2**][**2**])** **&&** **(**board**[**0**][**0**]** **==** 1**)** **&&** **(**board**[**1**][**1**]** **==** 0**))** // 대각선

**{**board**[**1**][**1**]** **=** 2**;** **break;}**

**else** **if((**board**[**0**][**2**]** **==** board**[**2**][**0**])** **&&** **(**board**[**0**][**2**]** **==** 1**)** **&&** **(**board**[**1**][**1**]** **==** 0**))** // 대각선

**{**board**[**1**][**1**]** **=** 2**;** **break;}**

**else**

**{**

**if((**board**[**i**][**0**]** **==** board**[**i**][**1**])** **&&** **(**board**[**i**][**1**]** **==** 1**)** **&** **(**board**[**i**][**2**]** **==** 0**))** // 가로

**{**board**[**i**][**2**]** **=** 2**;** **break;}**

**else** **if((**board**[**i**][**1**]** **==** board**[**i**][**2**])** **&&** **(**board**[**i**][**1**]** **==** 1**)** **&** **(**board**[**i**][**0**]** **==** 0**))** // 가로

**{**board**[**i**][**0**]** **=** 2**;** **break;}**

**else** **if((**board**[**i**][**0**]** **==** board**[**i**][**2**])** **&&** **(**board**[**i**][**0**]** **==** 1**)** **&** **(**board**[**i**][**1**]** **==** 0**))** // 가로

**{**board**[**i**][**1**]** **=** 2**;** **break;}**

**else** **if((**board**[**0**][**i**]** **==** board**[**1**][**i**])** **&&** **(**board**[**0**][**i**]** **==** 1**)** **&** **(**board**[**2**][**i**]** **==** 0**))** // 세로

**{**board**[**2**][**i**]** **=** 2**;** **break;}**

**else** **if((**board**[**1**][**i**]** **==** board**[**2**][**i**])** **&&** **(**board**[**1**][**i**]** **==** 1**)** **&** **(**board**[**0**][**i**]** **==** 0**))** // 세로

**{**board**[**0**][**i**]** **=** 2**;** **break;}**

**else** **if((**board**[**0**][**i**]** **==** board**[**2**][**i**])** **&&** **(**board**[**0**][**i**]** **==** 1**)** **&** **(**board**[**1**][**i**]** **==** 0**))** // 세로

**{**board**[**1**][**i**]** **=** 2**;** **break;}**

**else** **if(**i **>=** 2**)** // i가 끝에 도달했을때 (위의 for문이 i가 0부터 증가니깐, i가 2보다 크거나 같으면 끝임) 랜덤으로 아무데나 둔다

**{**

**while(**1**)**

**{**

xin **=** rand**()** **%** 3**;**

yin **=** rand**()** **%** 3**;**

**if(**board**[**yin**][**xin**]** **==** 0**)**

**{**board**[**yin**][**xin**]** **=** 2**;** **break;}**

**}**

**}**

**}**

**}**

**return** 0**;**

**}**

int WhoWin**(**void**)** // 마무리 출력 함수

**{**

**if(**win **==** 1**)**

printf**(**"\n\nO이 이겼습니다.\n"**);**

**else** **if(**win **==** 2**)**

printf**(**"\n\nX이 이겼습니다.\n"**);**

**else**

printf**(**"\n\n비겼습니다.\n"**);**

**return** 0**;**

**}**

int main**()**

**{**

/\* 플레이어가 O이다. \*/

srand**((**unsigned int**)** time**(NULL));**

**while(**1**)**

**{**

PrintBoard**();** // 보드판 출력 함수

printf**(**"O 차례 - y x 입력 : "**);**

scanf**(**"%d %d"**,** **&**y**,&**x**);**

**if(**board**[**y**][**x**]** **==** 0**)** // 보드가 비어있다면

board**[**y**][**x**]** **=** 1**;** // o를 놓는다.

/\* 검사부분 \*/

Check**();**

**if(**flag **==** 1**)**

**break;** // 나간다

flag **=** 1**;** // 승패가 아닌 비기는 경우를 위한 코드 (없으면 비길때 break가 안먹음 -> 승 패는 위의 코드에서 갈릴 것이고, 코드가 비어있다면 밑에 if문 때문에 flag가 0이 되서 계속 진행될 것이고, 코드가 다 찼지만 비긴다면 flag=1이니깐 break된다.)

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 0**)**

flag **=** 0**;** // flag = 1이면 빠져나온다

**}**

**}**

**if(**flag **==** 1**)**

**break;** // 나간다

/\* 인공지능 \*/

PrintBoard**();**

Ai**();**

/\* 검사부분 \*/

Check**();**

**if(**flag **==** 1**)**

**break;** // 나간다

flag **=** 1**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 0**)**

flag **=** 0**;** // flag = 1이면 빠져나온다

**}**

**}**

**if(**flag **==** 1**)**

**break;** // 나간다

**}**

/\* 마무리 부분 \*/

PrintBoard**();**

WhoWin**();**

**return** 0**;**

**}**

**2. 틱택토 아키텍처**

텍스트이(가) 표시된 사진

자동 생성된 설명

**3. 틱택토 게임을 두 사용자가 한 컴퓨터에서 플레이 할 수 있게 하기**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int board**[**3**][**3**]** **=** **{**0**,};**

int i**,** j**,** flag**,** win**,** xin**,** yin**,** x**,** y**;**

int PrintBoard**(**void**)** // 보드판 출력 함수

**{**

system**(**"clear"**);**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 1**)**

printf**(**"O |"**);**

**else** **if(**board**[**i**][**j**]** **==** 2**)**

printf**(**"X |"**);**

**else** **if(**board**[**i**][**j**]** **==** 0**)**

printf**(**" |"**);**

**}**

printf**(**"\n"**);**

**}** // 출력 부분

**return** 0**;**

**}**

int Check**(**void**)** // 검사 함수

**{**

flag **=** 0**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 가로

**{**

**if((**board**[**i**][**0**]** **==** board**[**i**][**1**])** **&&** **(**board**[**i**][**1**]** **==** board**[**i**][**2**]))**

**{**

**if(**board**[**i**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**i**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 세로

**{**

**if((**board**[**0**][**i**]** **==** board**[**1**][**i**])** **&&** **(**board**[**1**][**i**]** **==** board**[**2**][**i**]))**

**{**

**if(**board**[**1**][**i**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**i**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**0**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**2**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**2**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**0**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**return** 0**;**

**}**

int WhoWin**(**void**)** // 마무리 출력 함수

**{**

**if(**win **==** 1**)**

printf**(**"\n\nO이 이겼습니다.\n"**);**

**else** **if(**win **==** 2**)**

printf**(**"\n\nX이 이겼습니다.\n"**);**

**else**

printf**(**"\n\n비겼습니다.\n"**);**

**return** 0**;**

**}**

int main**()**

**{**

/\* 플레이어가 O이다. \*/

srand**((**unsigned int**)** time**(NULL));**

**while(**1**)**

**{**

PrintBoard**();** // 보드판 출력 함수

printf**(**"O 차례 - y x 입력 : "**);**

scanf**(**"%d %d"**,** **&**y**,&**x**);**

**if(**board**[**y**][**x**]** **==** 0**)** // 보드가 비어있다면

board**[**y**][**x**]** **=** 1**;** // o를 놓는다.

/\* 검사부분 \*/

Check**();**

**if(**flag **==** 1**)**

**break;** // 나간다

flag **=** 1**;** // 승패가 아닌 비기는 경우를 위한 코드 (없으면 비길때 break가 안먹음 -> 승 패는 위의 코드에서 갈릴 것이고, 코드가 비어있다면 밑에 if문 때문에 flag가 0이 되서 계속 진행될 것이고, 코드가 다 찼지만 비긴다면 flag=1이니깐 break된다.)

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 0**)**

flag **=** 0**;** // flag = 1이면 빠져나온다

**}**

**}**

**if(**flag **==** 1**)**

**break;** // 나간다

/\* 인공지능 \*/

PrintBoard**();**

printf**(**"X 차례 - y x 입력 : "**);**

scanf**(**"%d %d"**,** **&**y**,&**x**);**

**if(**board**[**y**][**x**]** **==** 0**)** // 보드가 비어있다면

board**[**y**][**x**]** **=** 2**;** // o를 놓는다.

/\* 검사부분 \*/

Check**();**

**if(**flag **==** 1**)**

**break;** // 나간다

flag **=** 1**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 0**)**

flag **=** 0**;** // flag = 1이면 빠져나온다

**}**

**}**

**if(**flag **==** 1**)**

**break;** // 나간다

**}**

/\* 마무리 부분 \*/

PrintBoard**();**

WhoWin**();**

**return** 0**;**

**}**

**4. server.c, client.c 분석 (간단한 문자열 전송 코드)**

**server.c**

#include <stdio.h>

#include <stdlib.h> //atoi를 사용하려면 있어야함

#include <string.h> // memset 등

#include <unistd.h> //sockaddr\_in, read, write 등

#include <arpa/inet.h> //htnol, htons, INADDR\_ANY, sockaddr\_in 등

#include <sys/socket.h> // socket 생성하려면 있어야함

void error\_handling**(**char **\*** message**);**

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

**{**

int serv\_sock**;**

int clnt\_sock**;**

/\* sockaddr\_in은 소켓 주소의 틀을 형셩해주는 구조체로 AF\_INET일 경우 사용 \*/

struct sockaddr\_in serv\_addr**;**

struct sockaddr\_in clnt\_addr**;** //accept함수에서 사용됨.

socklen\_t clnt\_addr\_size**;**

/\* TCP연결지향형이고 ipv4 도메인을 위한 소켓을 생성 \*/

serv\_sock**=**socket**(**PF\_INET**,** SOCK\_STREAM**,** 0**);**

**if(**serv\_sock **==** **-**1**)**

error\_handling**(**"socket error"**);**

/\* 주소를 초기화한 후 IP주소와 포트 지정 \*/

memset**(&**serv\_addr**,** 0**,** **sizeof(**serv\_addr**));**

serv\_addr**.**sin\_family**=**AF\_INET**;** //타입: ipv4

serv\_addr**.**sin\_addr**.**s\_addr**=**htonl**(**INADDR\_ANY**);** //ip주소

serv\_addr**.**sin\_port**=**htons**(**atoi**(**argv**[**1**]));** //port

/\* 소켓과 서버 주소를 바인딩 \*/

**if(**bind**(**serv\_sock**,** **(**struct sockaddr**\*)** **&**serv\_addr**,** **sizeof(**serv\_addr**))==-**1**)**

error\_handling**(**"bind error"**);**

/\* 연결 대기열 5개 생성 \*/

**if(**listen**(**serv\_sock**,** 5**)==-**1**)**

error\_handling**(**"listen error"**);**

/\* 클라이언트로부터 요청이 오면 연결 수락 \*/

clnt\_addr\_size **=** **sizeof(**clnt\_addr**);**

clnt\_sock**=**accept**(**serv\_sock**,** **(**struct sockaddr**\*)&**clnt\_addr**,** **&**clnt\_addr\_size**);**

**if(**clnt\_sock**==-**1**)**

error\_handling**(**"accept error"**);**

/\*-----데이터 전송-----\*/

char msg**[]** **=** "Hello this is server!\n"**;**

write**(**clnt\_sock**,** msg**,** **sizeof(**msg**));**

/\* 소켓들 닫기 \*/

close**(**clnt\_sock**);**

close**(**serv\_sock**);**

**return** 0**;**

**}**

void error\_handling**(**char **\***message**)**

**{**

fputs**(**message**,**stderr**);**

fputc**(**'\n'**,** stderr**);**

exit**(**1**);**

**}**

**client.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

void error\_handling**(**char**\*** message**);**

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

**{**

int clnt\_sock**;**

struct sockaddr\_in serv\_addr**;**

char message**[**1024**]** **=** **{**0x00**,** **};**

/\* TCP연결지향형이고 ipv4 도메인을 위한 소켓을 생성 \*/

clnt\_sock **=** socket**(**PF\_INET**,** SOCK\_STREAM**,** 0**);**

**if(**clnt\_sock **==** **-**1**)**

error\_handling**(**"socket error"**);**

/\* 인자로 받은 서버 주소 정보를 저장 \*/

memset**(&**serv\_addr**,** 0**,** **sizeof(**serv\_addr**));**

/\* 서버주소체계는 IPv4이다 \*/

serv\_addr**.**sin\_family **=** AF\_INET**;**

/\* 서버주소 IP저장해주기(인자로 받은거 넘겨주기) \*/

serv\_addr**.**sin\_addr**.**s\_addr **=** inet\_addr**(**argv**[**1**]);**

/\* 서버주소 포트번호 인자로 받은거 저장해주기 \*/

serv\_addr**.**sin\_port **=** htons**(**atoi**(**argv**[**2**]));**

/\* 클라이언트 소켓부분에 서버를 연결! \*/

**if(**connect**(**clnt\_sock**,** **(**struct sockaddr**\*)&**serv\_addr**,** **sizeof(**serv\_addr**))** **==** **-**1**)**

error\_handling**(**"connect error"**);**

/\* 연결이 성공적으로 되었으면 데이터 받기 \*/

**if(**read**(**clnt\_sock**,** message**,** **sizeof(**message**)-**1**)** **==** **-**1**)**

error\_handling**(**"read error"**);**

printf**(**"Message from server :%s\n"**,** message**);**

/\* 통신 후 소켓 클로우즈 \*/

close**(**clnt\_sock**);**

**return** 0**;**

**}**

void error\_handling**(**char**\*** message**)**

**{**

fputs**(**message**,** stderr**);**

fputc**(**'\n'**,** stderr**);**

exit**(**1**);**

**}**

**4-1. atoi**

atoi = char to int = 문자열을 정수 타입으로

\* 여기서 char는 char[N], char\*로 표현이 되는 문자열을 말한다.

**4-2. memset**

memset: 메모리의 내용(값)을 원하는 크기만큼 특정 값으로 세팅할 수 있는 함수

( memory + setting 메모리를 (특정 값으로) 세팅한다. )

\* 함수 원형

void \* memset(void \* ptr, int value, size\_t num);

첫번째 인자 void \* ptr은 세팅하고자 하는 메모리의 시작 주소이다. 즉, 그 주소를 가리키고 있는 포인터가 위치하는 자리이다.

두번째 인자 value는 메모리에 세팅하고자 하는 값을 집어넣으면 된다. int 타입으로 받지만 내부에서는 unsigned char로 변환되어서 저장된다. 즉 a 이런 것을 넣어도 무방하다는 뜻이다.

세번째 인자 size\_t num은 길이를 뜻한다. 이 길이는 바이트 단위로써 메모리의 크기 한 조각 단위의 길이를 말한다. 이는 보통 “길이 \* sizeof(데이터타입)”의 형태로 작성하면 된다.

반환 값은 성공하면 첫번째 인자로 들어간 ptr을 반환하고, 실패한다면 NULL을 반환한다.

**4-3. sockaddr, sockaddr\_in**

sockaddr 구조체는 소켓의 주소를 담는 기본 구조체의 역할을 한다.

struct sockaddr {

u\_short sa\_family; *// address family, 2 bytes*

char sa\_data[14]; *// IP address + Port number, 14 bytes*

};

sa\_family: 주소체계를 구분하기 위한 변수이며, 2bytes이다. 참고로, u\_short는 unsigned short를 말한다.

sa\_data: 실제 주소를 저장하기 위한 변수다. 14bytes이다.

즉, 이 구조체는 16bytes의 크기를 잡아준다.

sockaddr\_in은 sockaddr에서 sa\_family가 AF\_INET인 경우에 사용하는 구조체이다. sockaddr을 그대로 사용할 경우, sa\_data에 IP주소와 port번호가 조합되어 있어 쓰거나 읽기 불편하다. 그래서 sockaddr\_in을 사용한다. 이 구조체에서 사용하는 IP주소는 IPv4 주소체계를 사용한다.

struct sockaddr\_in { short sin\_family; *// 주소 체계: AF\_INET* u\_short sin\_port; *// 16 비트 포트 번호, network byte order* struct in\_addr sin\_addr; *// 32 비트 IP 주소* char sin\_zero[8]; *// 전체 크기를 16 비트로 맞추기 위한 dummy* }; struct in\_addr { u\_long s\_addr; *// 32비트 IP 주소를 저장 할 구조체, network byte order* };

sin\_family: 항상 AF\_INET을 설정한다. 이것은 필수이다.

sin\_port: 포트번호를 가진다. 2bytes이다. 즉, 포트번호는 0~65535의 범위를 갖는 숫자 값이다. 이 변수에 저장되는 값은 network byte order이어야 한다.

\* 참고로, 1024 이하의 포트 번호는 priviled port 혹은 reserver port라고 불리는 예약된 포트이다. 따라서, 권한을 가진 프로세스만이 1024 이하의 소켓주소에 대해서 바인딩(bind) 할 수 있다. 여기서 권환을 가진 프로세스라 함은, 리눅스에서는 CAP\_NET\_BIND\_SERVICE라는 capability를 가지고 있는 사용자 영역의 프로세스를 지칭한다.

\* 두번째 참고로, raw socket(socket type을 SOCK\_RAW로 생성한 소켓)은 포트 번호라는 개념을 가지지 않는다. 포트번호는 TCP 혹은 UDP에서만 구현하는 개념이다.

sin\_addr: 호스트 IP주소이다. 이 변수에는 INADDR\_ 로 시작하는 값, 예를 들면 INADDR\_ANY와 같은 것이 저장되어야 한다. 혹은 inet\_aton(), inet\_addr(), inet\_makeaddr()과 같은 라이브러리가 제공하는 함수의 반환값이 저장되어야 한다. 혹은 name resolver를 통해 직접 설정도 가능하다. 이 방법은 gethostbyname()을 알아보면 된다.

sin\_zero: 8bytes dummt data이다. 반드시 모두 0으로 채워져 있어야 한다. sockaddr\_in가 sin\_zero를 제외한 크기가 8bytes이므로, 총합이 16byte이다. struct sockaddr 구조체와 크기를 일치시키려는 목적인 걸 눈치 챌 수 있을 것이다. 여백을 채워서 16bytes로 만든다는 뉘앙스로 padding bytes 혹은 padding data라고도 한다.

**4-4. write**

write: open()함수로 열기를 한 파일에 쓰기를 한다.

\* 헤더: unistd.h

\* 형태: ssize\_t write (int fd, const void \*buf, size\_t n)

\* 인수: int fd 파일 디스크립터

void \*buf 파일에 쓰기를 할 내용을 담은 버퍼

size\_t n 쓰기할 바이트 개수

\* 반환: ssize\_t 쓰기에 성공했다면 쓰기한 바이트 개수를, 실패했다면 -1을 반환

**4-5. hton, ntoh**

hton, ntoh: 두 함수는 리틀 엔디안, 빅 엔디안을 사용하는 체계에서 빅 엔디안 또는 리틀 엔디안으로 변환해주는 함수이다. TCP/IP는 빅 엔디안 방식을 사용하므로 리틀 엔디안 체제에서 사용시 빅 엔디안으로의 자료 변환이 필요하다. 때문에 네트워크 소켓통신에서 빅 엔디안, 리틀 엔디안을 다뤄야 한다. 시스템마다 빅 엔디안 리틀 엔디안 사용이 다르지만 보통은 빅 엔디안을 사용한다. 하지만 리틀 엔디안을 사용하는 경우 빅 엔디안 형식으로의 변환이 필요하다.

\* 빅 엔디안: 시작 주소에 데이터의 최상위 비트가 오도록 저장

\* 리틀 엔디안: 시작 주소에 최하위 비트가 오도록 저장

htons 함수: 2byte(short형) 데이터에 대해 호스트 바이트 순서를 빅 엔디안 바이트 순서로 변환할 때 사용

htonl 함수: 4byte 데이터에 대해 호스트 바이트 순서를 빅 엔디안 바이트 순서로 변환할 때 사용

**4-6. INADDR\_ANY**

INADDR\_ANY: 자동으로 이 컴퓨터에 존재하는 랜카드 중 사용가능한 랜카드의 IP주소를 사용하라는 의미

예를들어 내 컴퓨터는 총 2개의 랜카드가 설치되어 있고 각각의 ip주소가 아래와 같다고 가정하자.

랜카드1: 192.168.0.100

랜카드2: 192.168.0.200

내 컴퓨터는 총 2개의 ip주소를 가지고 있으므로 외부에서 192.168.0.100으로 데이터를 보내건, 192.168.0.200으로 데이터를 보내건 내 컴퓨터로 오는 것은 똑같다. 하지만 프로그램은 다르다. 내가 만든 프로그램에 ip주소를 192.168.0.100으로 등록해 뒀다면 192.168.0.200으로 들어오는 데이터는 받지 않는다. 즉, 어차피 같은 컴퓨터임에도 불구하고 ip주소가 다르면 처리를 못한다는 의미이다. 이때, INADDR\_ANY를 사용하면 192.168.0.100으로 들어오건, 192.168.0.200으로 들어오건 모두 똑같이 처리할 수 있다.

**4-7. AF\_INET, PF\_INET**

AF\_INET: IP 주소 체계를 지칭할 때

PF\_INET: IP 자체를 가리킬 때

\* sockaddr\_in 구조체처럼 소켓의 주소와 함께 주소 체계를 지정하기 위해서는 AF\_INET를 사용해야 한다. AF는 Adresss Family의 줄임말이다.

\* socket() 함수처럼 실제 연결을 하기 위한 프로토콜을 지정하기 위해서는 PF\_INET를 사용해야 한다. PF는 Protocol Family의 줄임말이다.

**4-8. SOCK\_STREAM**

SOCK\_STREAM: tcp 타입의 소켓을 생성

**4-9. 주소 구조체 종류**

크게 주소를 담는 구조체는 sockaddr, sockaddr\_in, sockaddr\_in6, sockaddr\_un 등이 있다.

주소체계에 따른 소켓 구조체는 아래와 같다.

|  |  |
| --- | --- |
| **Socket Structure** | **Address Family** |
| struct sockaddr\_dl | AF\_LINK |
| struct sockaddr\_in | AF\_INET |
| struct sockaddr\_in6 | AF\_INET6 |
| struct sockaddr\_ll | AF\_PACKET |
| struct sockaddr\_un | AF\_UNIX |

1. sockaddr

sockaddr 구조체는 소켓의 주소를 담는 기본 구조체 틀의 역할을 한다. 그래서 보통 connect(연결요청)과 같은 함수들이 인자 타입으로 sockaddr을 받는다. sockaddr\_in으로 했던 sockaddr\_un으로 했던 결국 sockaddr 타입으로 형 변환한 값이 매개변수로 들어가게 된다.

\* sockaddr 구조체 구성

struct sockaddr {

u\_short sa\_family;

char sa\_data[14];

};

첫 번째 필드 sa\_family: 주소체계

두 번째 필드 sa\_data: 해당 주소체계에서 사용하는 주소 정보 (IP정보+포트정보)

2. sockaddr\_in

sockaddr\_in 구조체는 ipv4 주소를 저장하는 구조체이다. 현재 가장 많이 쓰이는 주소 체계이다.

\* sockaddr\_in 구조체 구성

struct sockaddr\_in {

sa\_family\_t sin\_family;

uint16\_t sin\_port;

struct in\_addr sin\_addr;

char sin\_zero[8];

};

첫 번째 필드 sin\_family: 주소체계를 저장하는 필드이다. sa\_family랑 변수명만 다를뿐 똑같다. sockaddr\_in은 ipv4를 위한 주소체계이니까 AF\_INET을 넣어주면 된다.

두 번째 필드 sin\_port: 포트 정보를 저장한다.

세 번째 필드 sin\_addr: IPv4 주소를 저장한다. 타입은 in\_addr 구조체이다.

struct in\_addr{

uint32\_t s\_addr; // 32비트 IPv4 주소

};

네 번째 필드 sin\_zero: 사용하지 않는 필드이다. 0으로 채워줘야 한다.

3. sockaddr\_in6

sockaddr\_in6 구조체는 IPv6 주소체계를 저장하기 위한 구조체이다.

\* sockaddr\_in6 구조체 구성

typedef struct sockaddr\_in6 {

sa\_family\_t sin6\_family;

in\_port\_t sin6\_port;

uint32\_t sin6\_flowinfo;

struct in6\_addr sin6\_addr;

uint32\_t sin6\_scope\_id;

}

첫 번째 필드 sin\_family: 주소체계를 저장하는 필드이다. sockaddr\_in6은 ipv6를 위한 주소체계이니까 AF\_INET6을 넣어주면 된다.

두 번째 필드 sin6\_port: 포트 정보를 저장한다.

세 번째 필드 sin6\_flowinfo:

텍스트이(가) 표시된 사진

자동 생성된 설명

32비트 필드로, IPv6 헤더 정보에서 찾을 수 있는 traffic class와 flow label를 포함하고 있다. 20bits전까지는 flow label, 그리고 20~27까지는 traffic class를 의미한다. 대게 이 필드는 0이다.

네 번째 필드 sin\_addr: IPv6 주소를 여기에 저장한다.

struct sin6\_addr {

uint8\_ts6\_addr[16];

};

IPv6는 128bit인거 아시죠? 8bit짜리 16개 해서 총 128bit를 저장한다.

다섯번째 필드 sin6\_scope\_id:

인터페이스 scope identifier

텍스트이(가) 표시된 사진

자동 생성된 설명

link-local 주소체계에서는 모든 인터페이스가 같은 네트워크를 가지고 있다. 그래서 어떤 인터페이스인지 구분할 필요가 있는데 그 값을 여기에 넣어준다. 특정 조건에서 사용되는 것이므로 대게 이 필드 값은 0이다.

**4-10. fputc, fputs**

int fputc(int, FILE \*); : 문자를 출력하기 위한 함수

int fputs(const char \*s, FILE \*stream); : 문자열을 출력하기 위한 함수

**4-11. sizeof**

size of : 메모리 공간에서 소모하는 메모리의 크기를 바이트 단위로 계산해서 반환하는 연산자

int main(void)

{

int num = 10;

int sz1 = sizeof(num); // 변수 num의 크기를 계산하여 sz1을 초기화

int sz2 = sizeof(int); // 자료형 int의 크기를 계산하여 sz2를 초기화

}

**4-12. argc, argv**

1. main 함수의 매개변수

C에서 main 함수는 프로그램 진입점이다. 즉, 최초로 함수로 구동되어 실행되는 시점이다. 함수의 매개변수는 이 함수를 호출할 때 전달해주는 정보를 뜻한다. 우리는 보통 함수를 만들어 놓고 호출할 때 매개변수에 정보를 넣어준다. 그런데 main 함수는 최초로 시작하는 진입점인데 어떻게 정보를 전달할까?

main 함수의 원형은 다음과 같다.

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

main 함수에 argc, argv라는 매개 변수를 전달한다.

int argc: 메인 함수에 전달되는 정보의 개수

char\* argv[]: 메인 함수에 전달되는 실질적인 정보로, 문자열의 배열을 의미

프로그램을 실행할 때 지정해 준 인자의 문자열들이 실제로 저장되는 배열

인덱스가 0인 문자열은 프로그램 실행경로로 항상 고정되어 있다.

2. main함수에 정보를 전달하는 원리

텍스트이(가) 표시된 사진

자동 생성된 설명

./arg I AM MINSUNG

argc에서 지속되는 인자는 ‘./arg’, ‘I’, ‘AM’, ‘MINSUNG’ 4개로 argv에는 문자열로 저장된 인자 주소값을 배열

형태로 저장한다. 즉, argv[0]에는 문자열 ‘arg\_test’의 시작 주소값이 저장되고 argv[1]에는 ‘I’의 시작주소 값이 담기게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3. 활용

- 파일 이름을 인자로 받아 해당 파일을 읽어오고 싶을 때

./test.exe input.txt로 프로그램을 실행시킨다면 argv[1]에 “input.txt”가 들어가게 되는데, 내가 인자로 넘겨준 파일을 읽어올 수 있도록 하는 것이다.

- 현재 프로그램이 개발 모드인지 프로덕션 모드인지 구분하고 싶을 때

개발을 하다보면 디버깅을 해야하는 부분이 있다. 매번 디버깅용 코드를 넣었다 프로덕션용으로 디플로이를 할 때 디버깅 코드를 지우면 귀찮으니 프로그램을 실행할 때 인자로 현재 개발모드라는 것을 알려주는 용도로 사용하는 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명



**5. 네트워크 프로그래밍을 이용해 틱택토 게임을 두 사용자가 서로 다른 컴퓨터에서 플레이 할 수 있게 하기**

**5-1. 실패 코드**

**tic\_ser.c**

#include <stdio.h>

#include <stdlib.h> //atoi를 사용하려면 있어야함

#include <string.h> // memset 등

#include <unistd.h> //sockaddr\_in, read, write 등

#include <arpa/inet.h> //htnol, htons, INADDR\_ANY, sockaddr\_in 등

#include <sys/socket.h> // socket 생성하려면 있어야함

void error\_handling**(**char **\*** message**);**

int board**[**3**][**3**]** **=** **{**0**,};**

int i**,** j**,** flag**,** win**,** xin**,** yin**,** x**,** y**;**

int PrintBoard**(**void**)** // 보드판 출력 함수

**{**

system**(**"clear"**);**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 1**)**

printf**(**"O |"**);**

**else** **if(**board**[**i**][**j**]** **==** 2**)**

printf**(**"X |"**);**

**else** **if(**board**[**i**][**j**]** **==** 0**)**

printf**(**" |"**);**

**}**

printf**(**"\n"**);**

**}** // 출력 부분

**return** 0**;**

**}**

int Check**(**void**)** // 검사 함수

**{**

flag **=** 0**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 가로

**{**

**if((**board**[**i**][**0**]** **==** board**[**i**][**1**])** **&&** **(**board**[**i**][**1**]** **==** board**[**i**][**2**]))**

**{**

**if(**board**[**i**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**i**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 세로

**{**

**if((**board**[**0**][**i**]** **==** board**[**1**][**i**])** **&&** **(**board**[**1**][**i**]** **==** board**[**2**][**i**]))**

**{**

**if(**board**[**1**][**i**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**i**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**0**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**2**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**2**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**0**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**return** 0**;**

**}**

int WhoWin**(**void**)** // 마무리 출력 함수

**{**

**if(**win **==** 1**)**

printf**(**"\n\nO이 이겼습니다.\n"**);**

**else** **if(**win **==** 2**)**

printf**(**"\n\nX이 이겼습니다.\n"**);**

**else**

printf**(**"\n\n비겼습니다.\n"**);**

**return** 0**;**

**}**

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

**{**

int serv\_sock**;**

int clnt\_sock**;**

/\* sockaddr\_in은 소켓 주소의 틀을 형셩해주는 구조체로 AF\_INET일 경우 사용 \*/

struct sockaddr\_in serv\_addr**;**

struct sockaddr\_in clnt\_addr**;** //accept함수에서 사용됨.

socklen\_t clnt\_addr\_size**;**

/\* TCP연결지향형이고 ipv4 도메인을 위한 소켓을 생성 \*/

serv\_sock**=**socket**(**PF\_INET**,** SOCK\_STREAM**,** 0**);**

**if(**serv\_sock **==** **-**1**)**

error\_handling**(**"socket error"**);**

/\* 주소를 초기화한 후 IP주소와 포트 지정 \*/

memset**(&**serv\_addr**,** 0**,** **sizeof(**serv\_addr**));**

serv\_addr**.**sin\_family**=**AF\_INET**;** //타입: ipv4

serv\_addr**.**sin\_addr**.**s\_addr**=**htonl**(**INADDR\_ANY**);** //ip주소

serv\_addr**.**sin\_port**=**htons**(**atoi**(**argv**[**1**]));** //port

/\* 소켓과 서버 주소를 바인딩 \*/

**if(**bind**(**serv\_sock**,** **(**struct sockaddr**\*)** **&**serv\_addr**,** **sizeof(**serv\_addr**))==-**1**)**

error\_handling**(**"bind error"**);**

/\* 연결 대기열 5개 생성 \*/

**if(**listen**(**serv\_sock**,** 5**)==-**1**)**

error\_handling**(**"listen error"**);**

/\* 클라이언트로부터 요청이 오면 연결 수락 \*/

clnt\_addr\_size **=** **sizeof(**clnt\_addr**);**

clnt\_sock**=**accept**(**serv\_sock**,** **(**struct sockaddr**\*)&**clnt\_addr**,** **&**clnt\_addr\_size**);**

**if(**clnt\_sock**==-**1**)**

error\_handling**(**"accept error"**);**

/\*-----데이터 전송-----\*/

**while(**1**)**

**{**

PrintBoard**();** // 보드판 출력 함수

printf**(**"O 차례 - y x 입력 : "**);**

scanf**(**"%d %d"**,** **&**y**,&**x**);**

**if(**board**[**y**][**x**]** **==** 0**)** // 보드가 비어있다면

board**[**y**][**x**]** **=** 1**;** // o를 놓는다.

/\* 검사부분 \*/

Check**();**

**if(**flag **==** 1**)**

**break;** // 나간다

flag **=** 1**;** // 승패가 아닌 비기는 경우를 위한 코드 (없으면 비길때 break가 안먹음 -> 승 패는 위의 코드에서 갈릴 것이고, 코드가 비어있다면 밑에 if문 때문에 flag가 0이 되서 계속 진행될 것이고, 코드가 다 찼지만 비긴다면 flag=1이니깐 break된다.)

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 0**)**

flag **=** 0**;** // flag = 1이면 빠져나온다

**}**

**}**

**if(**flag **==** 1**)**

**break;** // 나간다

write**(**clnt\_sock**,** board**,** **sizeof(**board**));**

**if(**read**(**clnt\_sock**,** board**,** **sizeof(**board**)-**1**)** **==** **-**1**)**

error\_handling**(**"read error"**);**

PrintBoard**();**

**}**

/\* 마무리 부분 \*/

PrintBoard**();**

WhoWin**();**

write**(**clnt\_sock**,** board**,** **sizeof(**board**));**

/\* 소켓들 닫기 \*/

close**(**clnt\_sock**);**

close**(**serv\_sock**);**

**return** 0**;**

**}**

void error\_handling**(**char **\***message**)**

**{**

fputs**(**message**,**stderr**);**

fputc**(**'\n'**,** stderr**);**

exit**(**1**);**

**}**

**tic\_cli.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/socket.h>

void error\_handling**(**char**\*** message**);**

int board**[**3**][**3**]** **=** **{**0**,};**

int i**,** j**,** flag**,** win**,** xin**,** yin**,** x**,** y**;**

int PrintBoard**(**void**)** // 보드판 출력 함수

**{**

system**(**"clear"**);**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 1**)**

printf**(**"O |"**);**

**else** **if(**board**[**i**][**j**]** **==** 2**)**

printf**(**"X |"**);**

**else** **if(**board**[**i**][**j**]** **==** 0**)**

printf**(**" |"**);**

**}**

printf**(**"\n"**);**

**}** // 출력 부분

**return** 0**;**

**}**

int Check**(**void**)** // 검사 함수

**{**

flag **=** 0**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 가로

**{**

**if((**board**[**i**][**0**]** **==** board**[**i**][**1**])** **&&** **(**board**[**i**][**1**]** **==** board**[**i**][**2**]))**

**{**

**if(**board**[**i**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**i**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 세로

**{**

**if((**board**[**0**][**i**]** **==** board**[**1**][**i**])** **&&** **(**board**[**1**][**i**]** **==** board**[**2**][**i**]))**

**{**

**if(**board**[**1**][**i**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**i**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**0**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**2**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**for(**i **=** 0**;** i **<** 3**;** i**++)** // 대각선

**{**

**if((**board**[**0**][**2**]** **==** board**[**1**][**1**])** **&&** **(**board**[**1**][**1**]** **==** board**[**2**][**0**]))**

**{**

**if(**board**[**1**][**1**]** **==** 1**)** // O가 이겼을 경우에

win **=** 1**,** flag **=** 1**;**

**if(**board**[**1**][**1**]** **==** 2**)** // X가 이겼을 경우에

win **=** 2**,** flag **=** 1**;**

**}**

**}**

**return** 0**;**

**}**

int WhoWin**(**void**)** // 마무리 출력 함수

**{**

**if(**win **==** 1**)**

printf**(**"\n\nO이 이겼습니다.\n"**);**

**else** **if(**win **==** 2**)**

printf**(**"\n\nX이 이겼습니다.\n"**);**

**else**

printf**(**"\n\n비겼습니다.\n"**);**

**return** 0**;**

**}**

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

**{**

int clnt\_sock**;**

struct sockaddr\_in serv\_addr**;**

char message**[**1024**]** **=** **{**0x00**,** **};**

/\* TCP연결지향형이고 ipv4 도메인을 위한 소켓을 생성 \*/

clnt\_sock **=** socket**(**PF\_INET**,** SOCK\_STREAM**,** 0**);**

**if(**clnt\_sock **==** **-**1**)**

error\_handling**(**"socket error"**);**

/\* 인자로 받은 서버 주소 정보를 저장 \*/

memset**(&**serv\_addr**,** 0**,** **sizeof(**serv\_addr**));**

/\* 서버주소체계는 IPv4이다 \*/

serv\_addr**.**sin\_family **=** AF\_INET**;**

/\* 서버주소 IP저장해주기(인자로 받은거 넘겨주기) \*/

serv\_addr**.**sin\_addr**.**s\_addr **=** inet\_addr**(**argv**[**1**]);**

/\* 서버주소 포트번호 인자로 받은거 저장해주기 \*/

serv\_addr**.**sin\_port **=** htons**(**atoi**(**argv**[**2**]));**

/\* 클라이언트 소켓부분에 서버를 연결! \*/

**if(**connect**(**clnt\_sock**,** **(**struct sockaddr**\*)&**serv\_addr**,** **sizeof(**serv\_addr**))** **==** **-**1**)**

error\_handling**(**"connect error"**);**

/\* 연결이 성공적으로 되었으면 데이터 받기 \*/

**while(**1**)**

**{**

**if(**read**(**clnt\_sock**,** board**,** **sizeof(**board**)-**1**)** **==** **-**1**)**

error\_handling**(**"read error"**);**

PrintBoard**();**

printf**(**"X 차례 - y x 입력 : "**);**

scanf**(**"%d %d"**,** **&**y**,&**x**);**

**if(**board**[**y**][**x**]** **==** 0**)** // 보드가 비어있다면

board**[**y**][**x**]** **=** 2**;** // o를 놓는다.

/\* 검사부분 \*/

Check**();**

**if(**flag **==** 1**)**

**break;** // 나간다

flag **=** 1**;**

**for(**i **=** 0**;** i **<** 3**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** 3**;** j**++)**

**{**

**if(**board**[**i**][**j**]** **==** 0**)**

flag **=** 0**;** // flag = 1이면 빠져나온다

**}**

**}**

**if(**flag **==** 1**)**

**break;** // 나간다

**}**

write**(**clnt\_sock**,** board**,** **sizeof(**board**));**

/\* 마무리 부분 \*/

**if(**read**(**clnt\_sock**,** board**,** **sizeof(**board**)-**1**)** **==** **-**1**)**

error\_handling**(**"read error"**);**

PrintBoard**();**

WhoWin**();**

/\* 통신 후 소켓 클로우즈 \*/

close**(**clnt\_sock**);**

**return** 0**;**

**}**

void error\_handling**(**char**\*** message**)**

**{**

fputs**(**message**,** stderr**);**

fputc**(**'\n'**,** stderr**);**

exit**(**1**);**

**}**

**5-2. 실패 원인이 무엇일까?**

초반에는 송신이 잘되다가 1, 2번 송수신한 후 송신도 안 되고 수신도 안된다. 공부한 server.c와 client.c를 이용하여 중간에 tictactoe.c를 집어넣어 코드를 완성시키는 방식으로 하였는데, 뭔가 빠진 부분과 부족한 부분이 있었던 것 같다. 아직까지 정확한 실패원인을 몰라서 해결을 못하고 있다.