KNU TALK 프로그램 보고서

목차

[KNU TALK 프로그램 보고서 1](#_Toc74336479)

[개요 3](#_Toc74336480)

[Server(완) 3](#_Toc74336481)

[헤더파일(완) 4](#_Toc74336482)

[Server\_header.h(완) 4](#_Toc74336483)

[개요(완) 4](#_Toc74336484)

[전처리(완) 4](#_Toc74336485)

[배열 길이 지정(완) 4](#_Toc74336486)

[데이터 구조체(완) 4](#_Toc74336487)

[서버에서 유지하는 데이터(완) 4](#_Toc74336488)

[파일 및 뮤텍스(완) 4](#_Toc74336489)

[오류코드 메세지 배열(완) 4](#_Toc74336490)

[함수 원형(완) 4](#_Toc74336491)

[Packet\_header.h(완) 4](#_Toc74336492)

[개요(완) 4](#_Toc74336493)

[오류 코드(완) 4](#_Toc74336494)

[패킷 타입 지정(완) 4](#_Toc74336495)

[공통요소(완) 4](#_Toc74336496)

[가입 패킷(완) 4](#_Toc74336497)

[로그인 패킷(완) 4](#_Toc74336498)

[로그아웃 패킷(완) 4](#_Toc74336499)

[방 입장 패킷(완) 4](#_Toc74336500)

[방 퇴장 패킷(완) 4](#_Toc74336501)

[채팅 패킷(완) 4](#_Toc74336502)

[차단 패킷(완) 4](#_Toc74336503)

[정보요청 패킷(완) 4](#_Toc74336504)

[방 생성 패킷(완) 4](#_Toc74336505)

[소스파일(완) 4](#_Toc74336506)

[Main.c(완) 4](#_Toc74336507)

[Thread.c(완) 4](#_Toc74336508)

[Accept\_thread(완) 4](#_Toc74336509)

[WorkerThread(완) 4](#_Toc74336510)

[Packet\_handler.c(완) 4](#_Toc74336511)

[Packet\_construct(완) 4](#_Toc74336512)

[Packet\_handler(완) 4](#_Toc74336513)

[Packet\_send(완) 4](#_Toc74336514)

[User\_interaction.c(완) 4](#_Toc74336515)

[Login(완) 4](#_Toc74336516)

[Logout(완) 4](#_Toc74336517)

[Member\_register(완) 4](#_Toc74336518)

[Registered\_user(완) 4](#_Toc74336519)

[Search\_user(완) 4](#_Toc74336520)

[DB\_management.c(완) 4](#_Toc74336521)

[개요(완) 4](#_Toc74336522)

[파일 이름(완) 4](#_Toc74336523)

[파일데이터 구조체(완) 4](#_Toc74336524)

[Init\_server(완) 4](#_Toc74336525)

[Write\_to\_file(완) 4](#_Toc74336526)

[read\_from\_file(완) 4](#_Toc74336527)

[Room\_management.c(완) 4](#_Toc74336528)

[Make\_room(완) 4](#_Toc74336529)

[Enter\_room(완) 4](#_Toc74336530)

[Quit\_room(완) 4](#_Toc74336531)

[Delete\_room(완) 4](#_Toc74336532)

[Room\_info\_request(완) 4](#_Toc74336533)

[Current\_room\_num(완) 4](#_Toc74336534)

[Find\_empty\_room(완) 4](#_Toc74336535)

[Chat\_processor.c(완) 4](#_Toc74336536)

[Add\_block\_list(완) 4](#_Toc74336537)

[Echo\_message(완) 4](#_Toc74336538)

[Error\_handling.c(완) 4](#_Toc74336539)

[Client(완) 4](#_Toc74336540)

[헤더파일(완) 4](#_Toc74336541)

[Client\_header.h(완) 4](#_Toc74336542)

[전처리(완) 4](#_Toc74336543)

[UI (완) 4](#_Toc74336544)

[Cursor(완) 4](#_Toc74336545)

[오류코드(완) 4](#_Toc74336546)

[데이터 구조체(완) 4](#_Toc74336547)

[오류코드 메시지 배열(완) 4](#_Toc74336548)

[전역변수(완) 4](#_Toc74336549)

[함수 원형(완) 4](#_Toc74336550)

[Cln\_packet\_header.h(완) 4](#_Toc74336551)

[소스파일 4](#_Toc74336552)

[Main.c(완) 4](#_Toc74336553)

[Main\_thread.c(완) 4](#_Toc74336554)

[Packet\_handler.c(완) 4](#_Toc74336555)

[Packet\_construct(완) 4](#_Toc74336556)

[Packet\_handler(완) 4](#_Toc74336557)

[Packet\_send(완) 4](#_Toc74336558)

[UI.c(완) 4](#_Toc74336559)

[Init\_console(완) 4](#_Toc74336560)

[Set\_console\_size(완) 4](#_Toc74336561)

[Title(완) 4](#_Toc74336562)

[Lobby(완) 4](#_Toc74336563)

[Login(완) 4](#_Toc74336564)

[Reg\_session(완) 4](#_Toc74336565)

[Chat-window(완) 4](#_Toc74336566)

[Move\_cursor(완) 4](#_Toc74336567)

[Set\_cursor\_view(완) 4](#_Toc74336568)

[Print\_on\_xy(완) 4](#_Toc74336569)

[Move\_to\_xy(완) 4](#_Toc74336570)

[Error\_handling.c(완) 4](#_Toc74336571)

# 개요

본 파일은 IOCP 기반 채팅 프로그램의 소스코드에 대한 상세한 설명이 담겨있는 문서입니다.

각 목차나 내용에 있는 링크를 CLTR 키를 누른 채 클릭하면 해당 내용으로 바로 이동하실 수 있습니다.

코드가 방대해 사진이 작게 보일 수 있으니 소스코드와 같이 보길 권장합니다.

컴파일 환경은 Visual Studio 2019 이며 낮은 버전 사용시 컴파일 에러가 발생할 수 있습니다.

# Server(완)

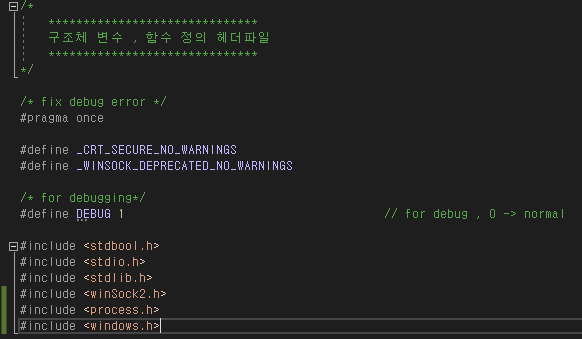
## **헤더파일(완)**

### Server\_header.h(완)

#### 개요(완)

Server\_header.h에는 헤더파일, 전처리 명령어, 구조체, 함수 원형, 전역변수들이 담겨있습니다.

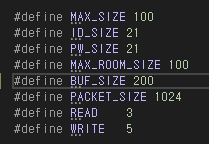
#### 전처리(완)



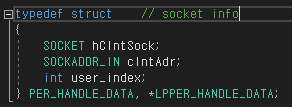
서버 가동에 필요한 기본적인 전처리 지시자입니다.

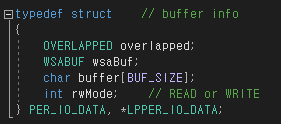
DEBUG를 1로 설정하면 서버 실행 시 각종 함수 실행 로그를 출력하게끔 했습니다.

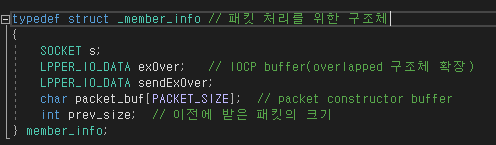
#### 배열 길이 지정(완)

각 배열의 길이를 따로 지정해 주었습니다.

#### 데이터 구조체(완)

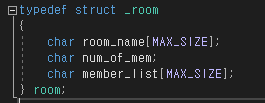
클라이언트 소켓, 주소정보가 담겨있는 구조체입니다. User\_index를 추가로 만들어 각종 데이터를 user\_index로 쉽게 접근할 수 있게 했습니다.

iocp에서 사용되는 구조체입니다.

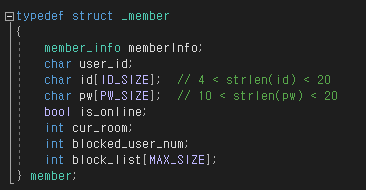


각 클라이언트로부터 받은 패킷 처리를 위한 구조체입니다.

클라이언트 소켓 정보, iocp 버퍼 2개, packet 처리를 위한 버퍼, 미처 처리하지 못한 패킷이 있다면 재조립을 위해 사이즈를 저장할 변수가 있습니다. (처리가 한 번에 완료되었다면 prev\_size 변수에는 0이 들어있습니다.)

방을 나타내는 구조체로서 각 방은 방의 이름,

방의 인원수, 방에 있는 유저 목록을 유지합니다.

멤버를 나타내는 구조체입니다.

MemberInfo = 연결시 소켓정보 저장

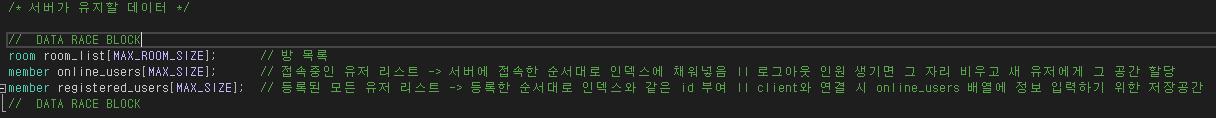
Is\_online = 유저의 접속여부

Cur\_room = 현재 채팅중인 방의 id

Blocked\_user\_num = 차단리스트에 등록된 유저 수

Block\_list = 차단 유저 목록

#### 서버에서 유지하는 데이터(완)



Room\_list = 생성된 방 목록

Online\_users = 서버에 접속중인 클라이언트 목록

Registered\_users = 서버에 가입한 유저 목록

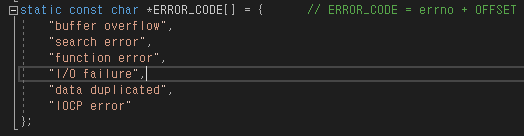
서버 시작 시 기존 데이터를 읽어들이고, 종료시 갱신된 목록을 파일에 새로이 저장합니다.

#### 파일 및 뮤텍스(완)



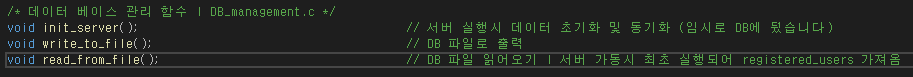
파일 및 뮤텍스 변수입니다.

#### 오류코드 메세지 배열(완)

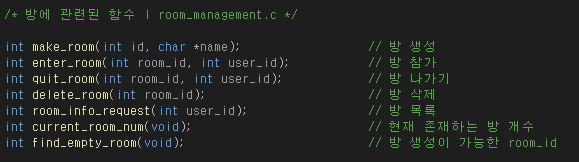


각 오류 값에 해당하는 메시지를 쉽게 출력하기위해 만든 배열입니다.

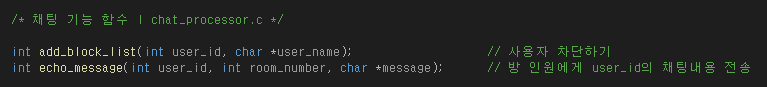
#### 함수 원형(완)



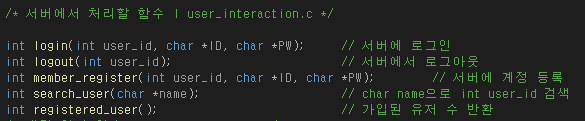
데이터 베이스 관리 함수입니다. [DB\_management.c](#_DB_management.c(완))에 정의되어 있습니다.



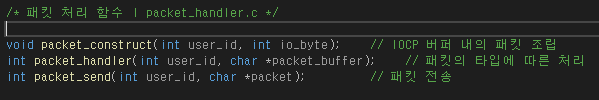
방(room)과 관련된 함수들의 원형입니다. [Room\_management.c](#_Room_management.c)에 정의되어 있습니다.



채팅 기능 함수입니다. 차단 기능, 에코 기능 두가지가 있으며 [chat\_processor.c](#_Chat_processor.c)에 정의되어 있습니다.



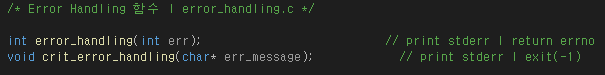
서버에서 처리를 담당할 함수의 원형들입니다. [user\_interaction.c](#_User_interaction.c(완))에 정의되어 있습니다.



패킷을 처리하는 함수입니다. [Packet\_handler.c](#_Packet_handler)에 정의되어 있습니다.



스레드 관련 함수입니다. [Thread.c](#_Thread.c)에 정의되어 있습니다.



에러를 처리하는 함수입니다. [Error\_handling.c](#_Error_handling.c(완))에 정의되어 있습니다.

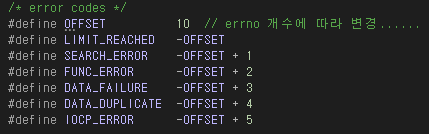
### Packet\_header.h(완)

#### 개요(완)

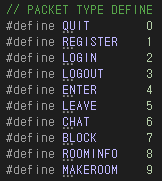
채팅 패킷과 방 퇴장 패킷을 제외한 모든 패킷은 클라이언트가 요청 후 서버로부터 응답 패킷을 받는 1vs1 형식으로 되어있습니다.

채팅 패킷과 방 퇴장 패킷은 응답 패킷이 없고, 서버가 여러 유저에게 그 내용을 전달하는 1vs다 형식입니다.

#### 오류 코드(완)

각 패킷 전송시 전달될 오류 코드들을 만들었습니다.

#### 패킷 타입 지정(완)



패킷의 종류를 구별하기 위해 각각의 타입을 만들어 주었습니다.

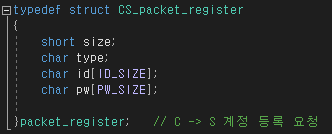
#### 공통요소(완)

모든 패킷은 기본적으로 패킷의 크기와 타입을 저장합니다.

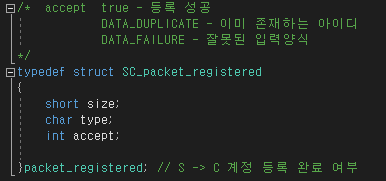
또한 클라이언트가 서버에 요청을 보내는 패킷은 C -> S

서버가 클라이언트에게 응답해주는 패킷은 S -> C 로 표기했습니다.

#### 가입 패킷(완)

등록하고자 하는 id, pw를 서버에 전달합니다

.

가입성공 여부를 전달하는 패킷입니다.

Accept 변수의 값으로는

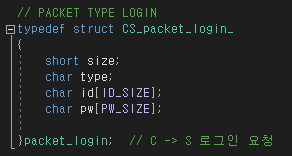
True = 성공

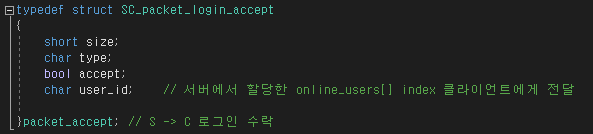
DATA\_DUPLICATE (id 중복)

DATA\_FAILURE (id, pw 길이제한 불만족)

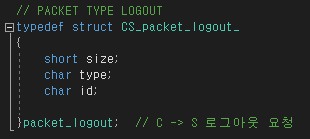
이 있습니다.

#### 로그인 패킷(완)

id, pw를 서버에 전달해 로그인을 요청합니다.

  
로그인 성공여부와 성공 시 부여받을 user\_id를 전달합니다.

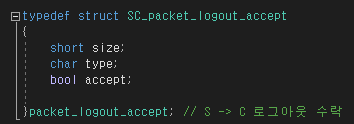
#### 로그아웃 패킷(완)

로그아웃을 요청하는 패킷입니다.

서버로부터 부여받은 id를 전달해 어떤

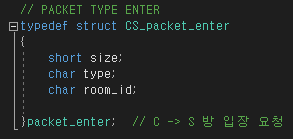
유저가 로그아웃을 요청하는지 식별케

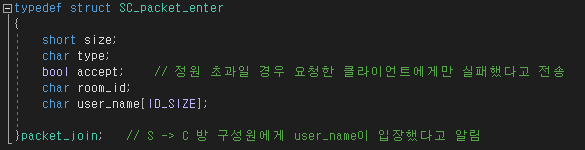
했습니다.



로그아웃 성공 여부를 전달합니다.

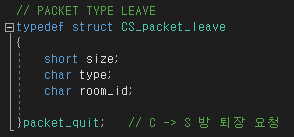
#### 방 입장 패킷(완)

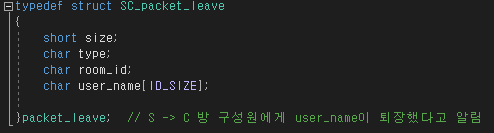
입장하고자 하는 방의 id를 서버에 전달합니다.



방에있던 기존의 유저들에게 새 유저(user\_name) 가 입장했음을 알리는 패킷입니다.

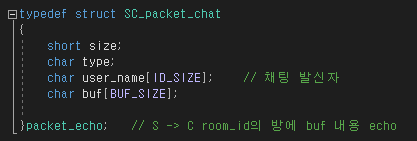
#### 방 퇴장 패킷(완)

퇴장하고자 하는 방의 id를 서버에 전달합니다.

  
방에있던 기존의 유저들에게 새 유저(user\_name) 가 퇴장했음을 알리는 패킷입니다.

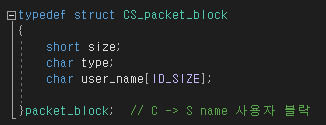
#### 채팅 패킷(완)

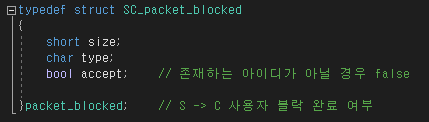
채팅내용과 그것을 전달할 방의 id를 전달합니다.



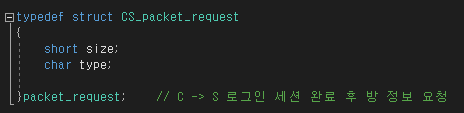
방의 다른 유저들에게 발신자 id 및 채팅내용을 전달합니다.

#### 차단 패킷(완)

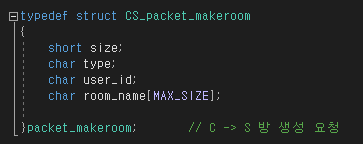
user\_name을 전달해 해당하는 유저를 차단목록에 추가 할것을 요청하는 패킷입니다.

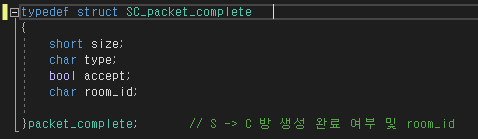
차단목록 추가 성공 여부를 받습니다.

#### 정보요청 패킷(완)

로그인 세션 완료, 혹은 새로고침을할 때마다 방 정보의 갱신을 요구하는 패킷입니다.

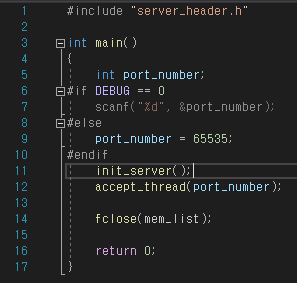
#### 방 생성 패킷(완)

이름을 전달받아 해당하는 이름의 방을 생성합니다.

방 생성 완료 여부를 전달합니다. 성공 시 할당된 방의 id를 전달합니다.

## **소스파일(완)**

### Main.c(완)

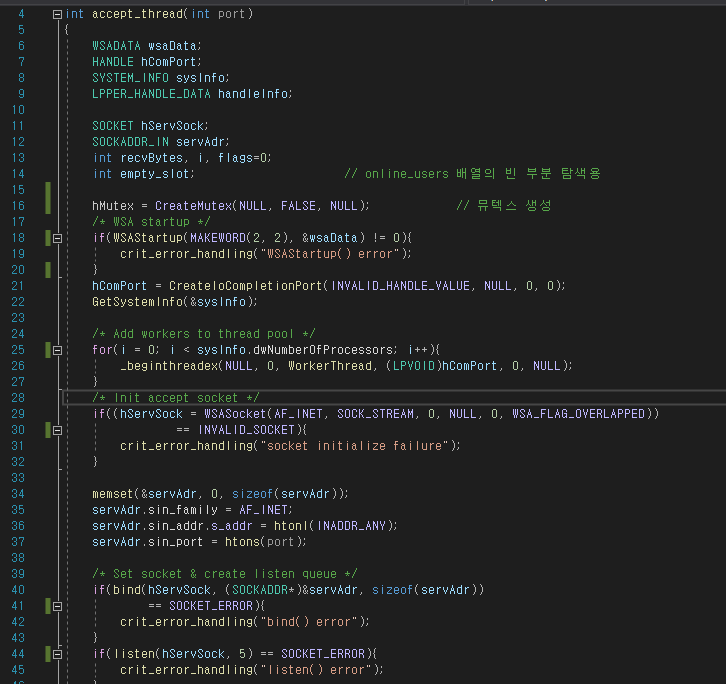
main 에서는 사용자로부터 port 번호를 입력 받아서 accept\_thread를 생성합니다.

Init\_server 함수는 서버 시작 전 데이터 초기화 작업을 하는 함수입니다.

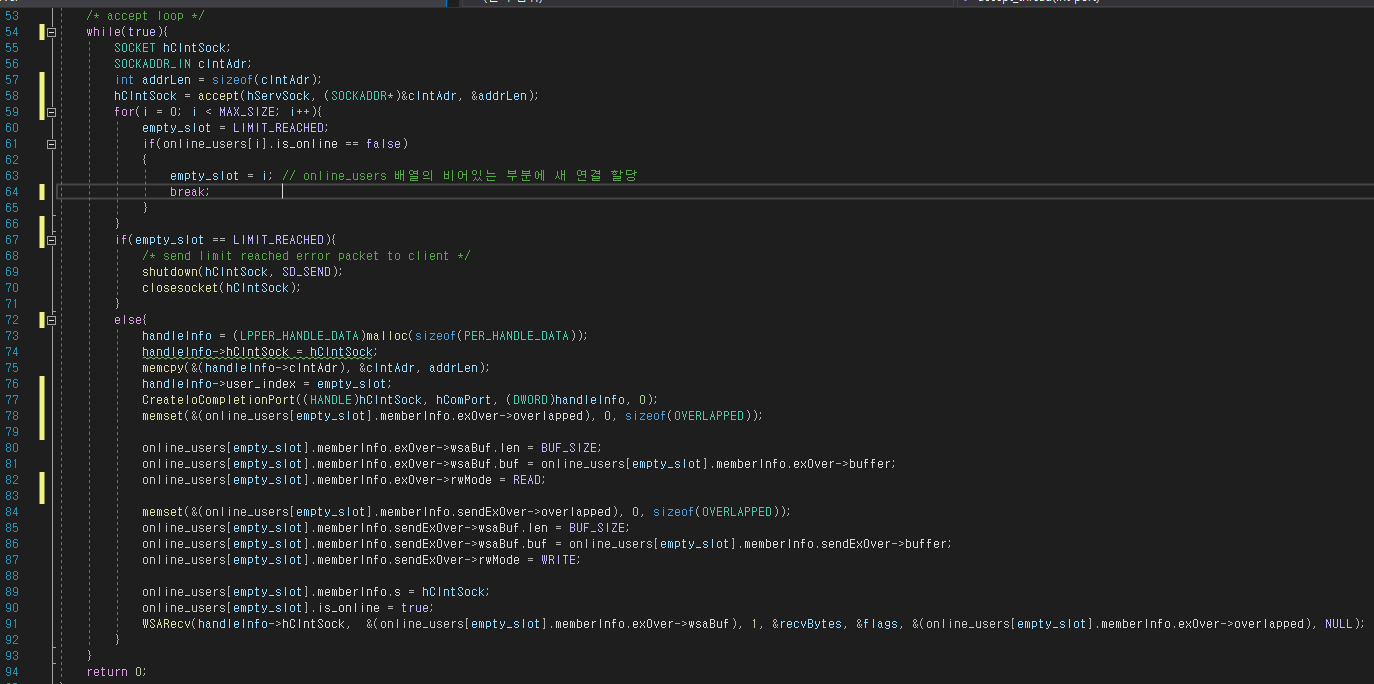
디버깅 모드에선 포트를 고정시켜 실행시마다 입력하는 수고를 제거하였습니다.

### Thread.c(완)

#### Accept\_thread(완)



Accept\_thread의 첫 부분입니다. 뮤텍스 생성, Iocp 서버 가동에 필요한 CompletionPort 생성, 스레드 생성, 서버 소켓을 생성해 bind, listen 함수를 실행하는 모습을 볼 수 있습니다.

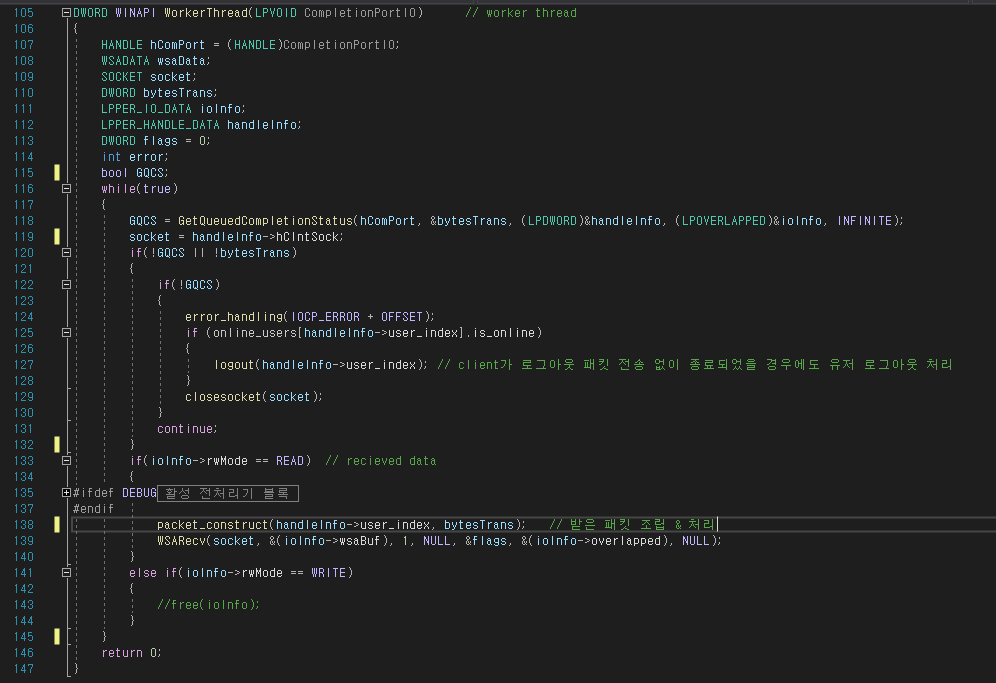


두 번째 부분은 accept loop입니다.

클라이언트가 연결요청을 하면 online\_users 배열의 빈 곳에 클라이언트의 overlapped와 소켓 정보를 저장합니다.

접속가능한 최대 인원을 초과했다면 연결 요청을 받지 않습니다.

#### WorkerThread(완)



WorkerThread에서는 IO가 완료되면 CP오브젝트의 정보를 읽고 packet\_construct 함수를 호출해 패킷을 조립합니다. 이후 다시 Recv로 전환해 패킷을 기다립니다. 클라이언트가 로그아웃 과정을 거치지 않고 비정상적으로 종료할 경우 로그아웃 처리를 별도로 해줍니다.

### Packet\_handler.c(완)

#### Packet\_construct(완)



전달받은 데이터를 패킷으로 만드는 함수입니다. 처리 전, 이전에 처리하지 못한 데이터가 있다면 그 크기를 변수에 저장합니다. (= prev\_size)

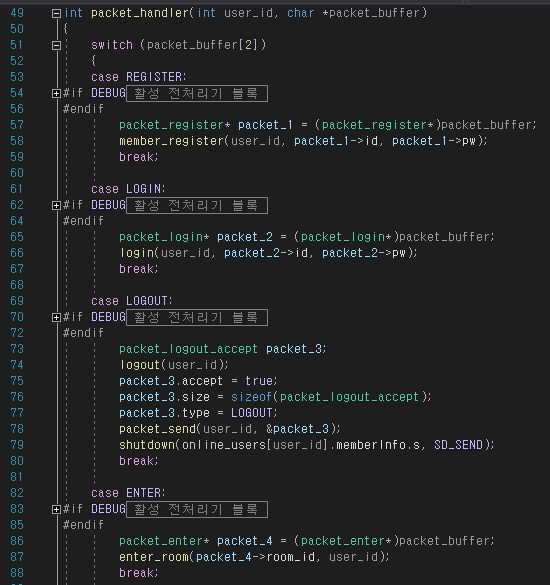
패킷의 생성은 우선 전달받은 패킷의 사이즈를 저장해 그 데이터로 패킷 완성이 가능한지 판별합니다.

만약 가능하다면 패킷을 보낸 유저의 packet\_buf에 패킷의 크기만큼 데이터를 그대로 복사하고 처리해야할 데이터(rest\_byte) 및 패킷이 담긴 버퍼의 주소(= p)를 갱신합니다.

남은 데이터로 패킷 완성이 불가능하면 패킷을 보낸 유저의 packet\_buf에 임시로 저장해두어 다음에 처리하도록 합니다.

위 과정을 처리해야할 데이터 크기(= rest\_byte)가 0보다 작아질 때까지 반복합니다.

#### Packet\_handler(완)

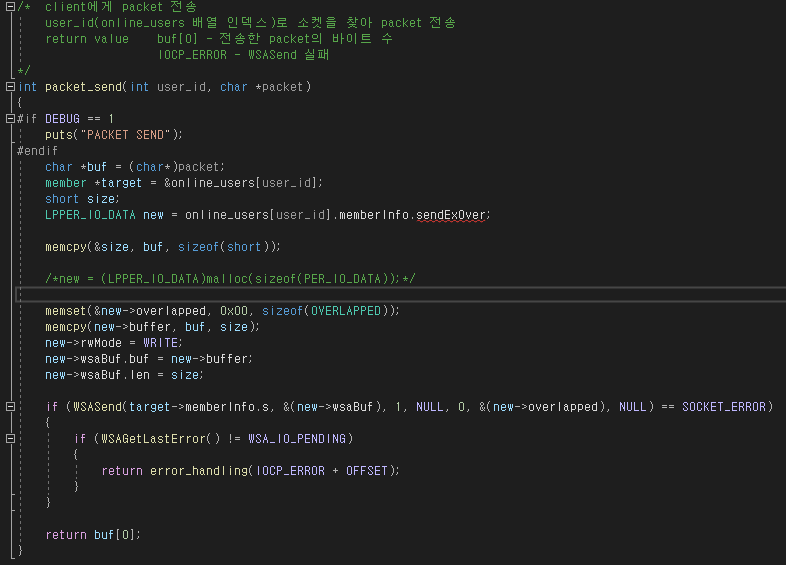




전달받은 패킷의 type을 참조하여 해당하는 패킷 형태로 변환 후 정해진 함수를 실행합니다.

디버그 모드에선 실행되는 어떠한 기능을 요청한 유저의 id와 그 패킷의 타입을 출력합니다.

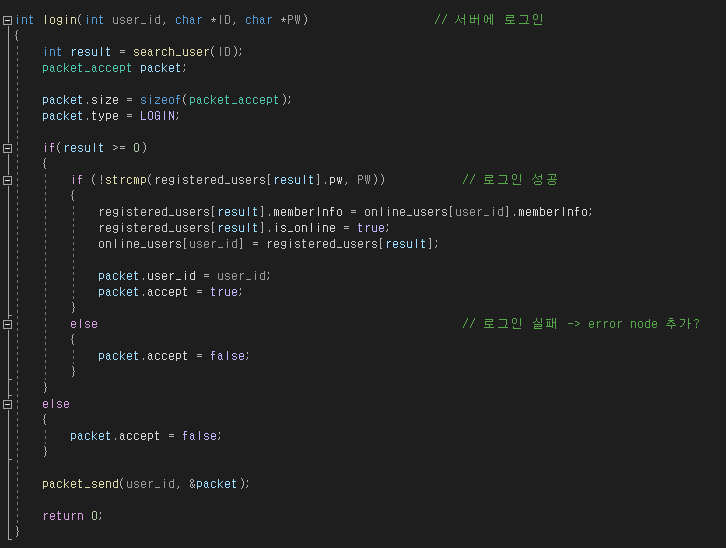
#### Packet\_send(완)



서버에서 클라이언트에 패킷을 보내는 함수입니다. Online\_users 배열을 참조하여 어떤 클라이언트에게 보낼지 식별한 후(new에 저장) 패킷을 전송합니다. 오류 발생시 오류메시지를 출력합니다..

### User\_interaction.c(완)

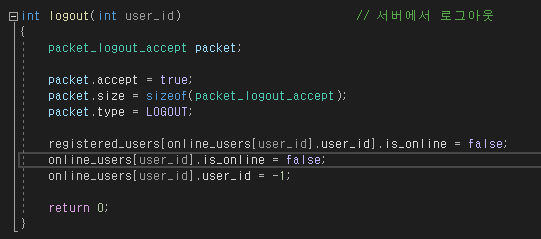
#### Login(완)



가입된 유저목록에서 id, pw 일치 여부를 확인 후 online\_users 배열에 추가하고 로그인 성공 여부가 담긴 패킷을 클라이언트에 전달합니다.

또한 is\_online을 true로 설정해 현재 접속중임을 나타냅니다.

#### Logout(완)



User\_id를 받아 is\_online을 false 로 바꿔주고 online\_users 배열에서 해당하는 곳을 다시 사용할 수 있게 -1로 바꾸어 로그아웃 처리를 합니다.

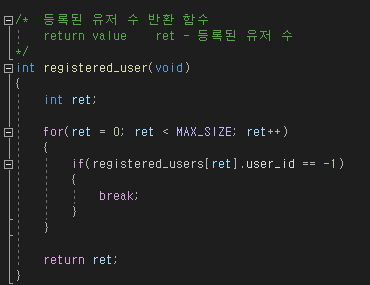
#### Member\_register(완)



Id, pw를 받아 길이 만족 여부 및 중복 여부를 검사 후 생성을 성공하면 registered\_users 배열에 새로 추가하고

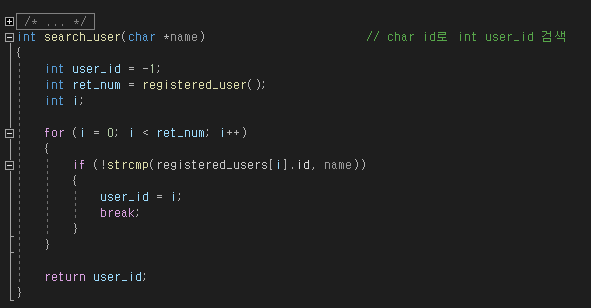
결과 패킷을 만들어 클라이언트에게 보냅니다.

#### Registered\_user(완)

등록된 유저의 수를 반환하는 함수입니다.

빈자리를 -1로 초기화했기 때문에 -1을 만나면 그 index를 반환합니다.

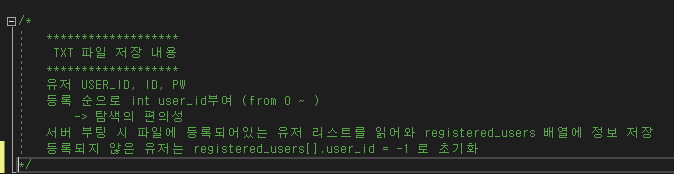
#### Search\_user(완)



전달받은 id로 해당하는 유저가 있는지 검사합니다. 있을 경우 그 유저의 index, 없을 경우 -1을 반환합니다.

### DB\_management.c(완)

#### 개요(완)

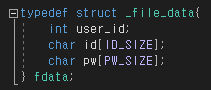


DB\_management의 기본적인 설명 주석입니다.

#### 파일 이름(완)

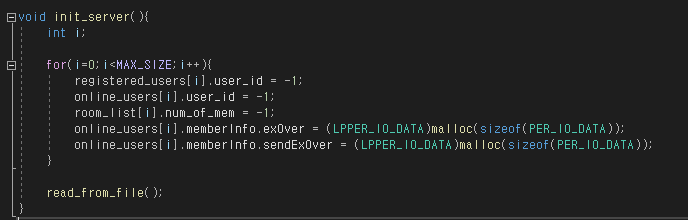
 파일의 이름입니다.

#### 파일데이터 구조체(완)

주석에서 언급한 파일에 저장될 내용을

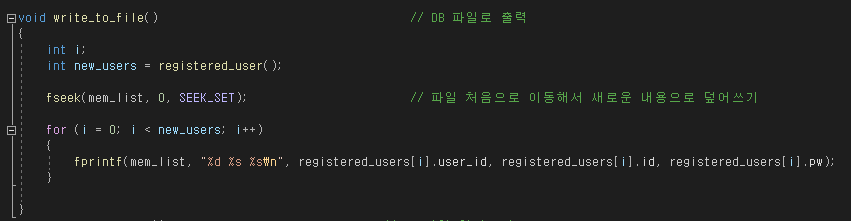
구조체로 만들었습니다.

#### Init\_server(완)

서버 실행에 필요한 각종 데이터를 초기화 합니다.

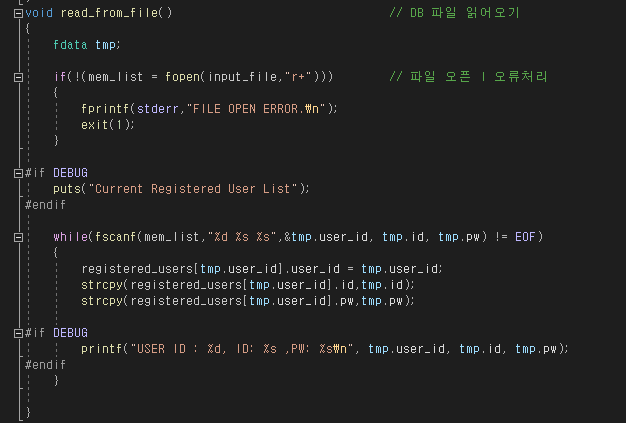
또한 read\_from\_file 함수를 호출해 가입 한 유저 배열(registered\_users)을 갱신합니다.

#### Write\_to\_file(완)



서버 실행 중 갱신된 registered\_users 배열을 파일에 저장합니다.

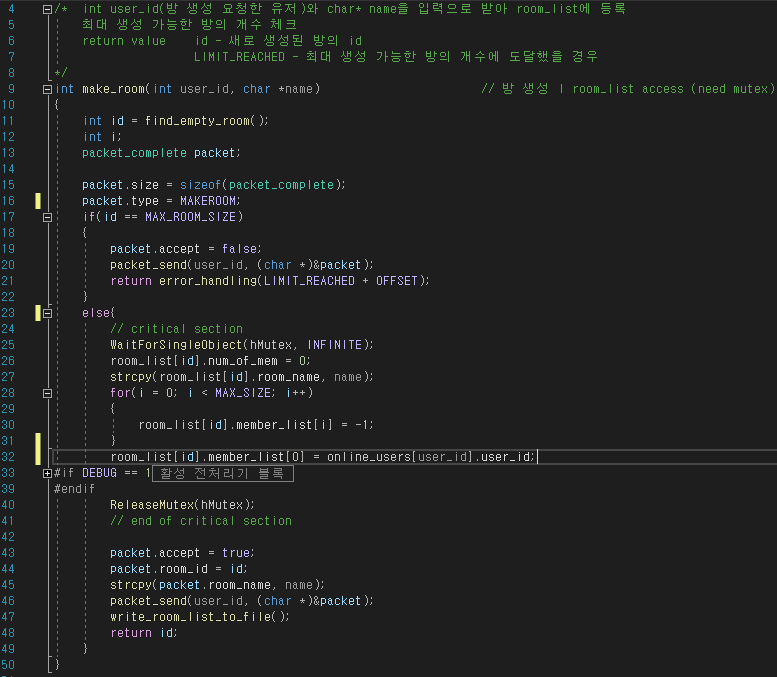
#### read\_from\_file(완)

파일로부터 가입한 유저 정보를 읽어 registered\_users 배열에 저장합니다.

디버그 모드에선 가입 유저 목록을 출력합니다.

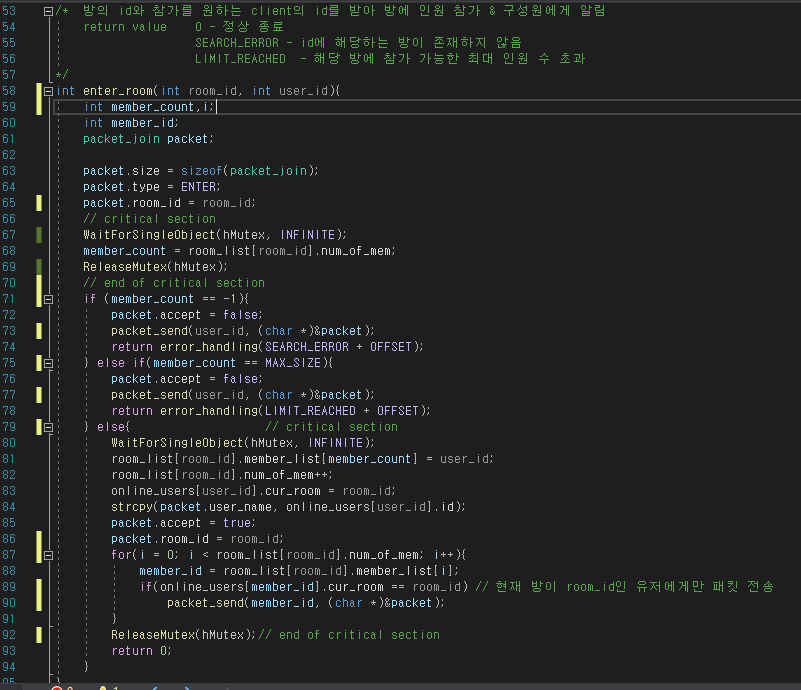
### Room\_management.c(완)

#### Make\_room(완)



User\_id와 name을 받아 해당하는 name의 방을 생성합니다. 생성 후 멤버 수를 0, 이름을 name, member\_list 배열을 -1로 초기화 해주고 생성한 유저를 해당 방에 참여자로 등록합니다. 이후 새로 생성된 방의 정보를 파일의 끝에 추가합니다.

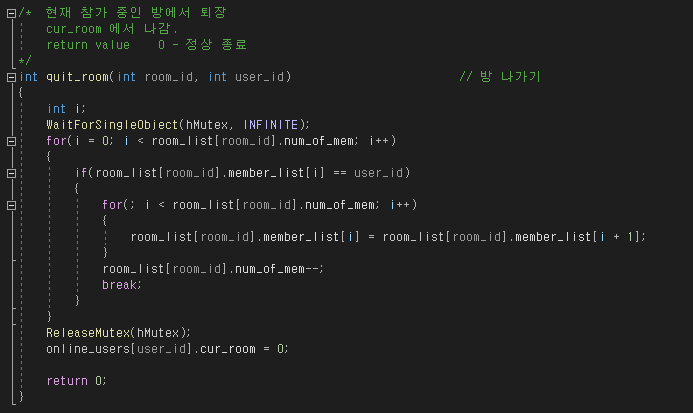
#### Enter\_room(완)



User\_id, room\_id를 받아 방의 Member\_list에 유저 등록 및 num\_of\_mem++를 통해 해당 방의 유저로 등록합니다. 이후 그 방 유저 모두에게 방 입장 패킷을 전송해 유저의 접속 사실을 알립니다.

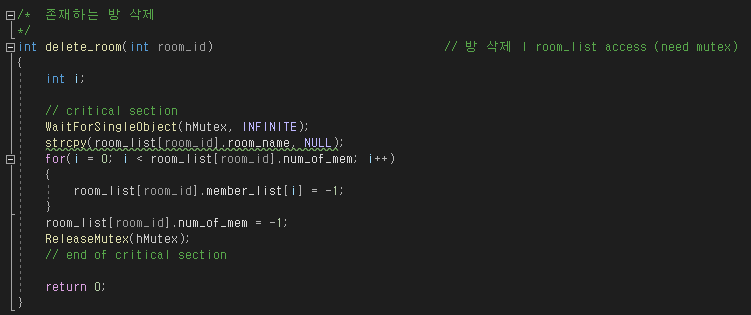
Member\_count 가 -1 혹은 MAX\_SIZE 면 방 입장에 실패하고 결과 패킷을 해당 유저에게 전송합니다.

#### Quit\_room(완)



User\_id와 room\_id를 받아 해당하는 유저를 그 방에서 제외합니다. 해당하는 유저의 정보를 member\_list를 compaction 해주어 유저 정보를 지우고 방의 멤버 수를 하나 줄입니다.

#### Delete\_room(완)

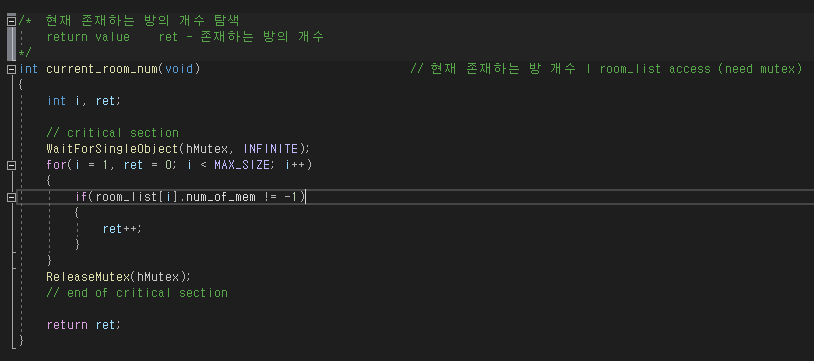


Room\_id를 통해 해당하는 방을 삭제합니다. 방의 정보를 -1로 초기화해 다시 사용할 수 있게 합니다.

#### Room\_info\_request(완)

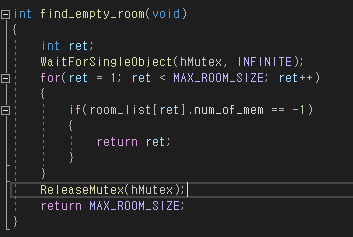
  
방의 정보를 패킷으로 만들어 클라이언트에게 보내는 패킷입니다. 방의 이름, 방에 있는 유저 수를 전달합니다.

#### Current\_room\_num(완)



현재 서버내에 존재하는 방의 개수를 반환하는 함수입니다.

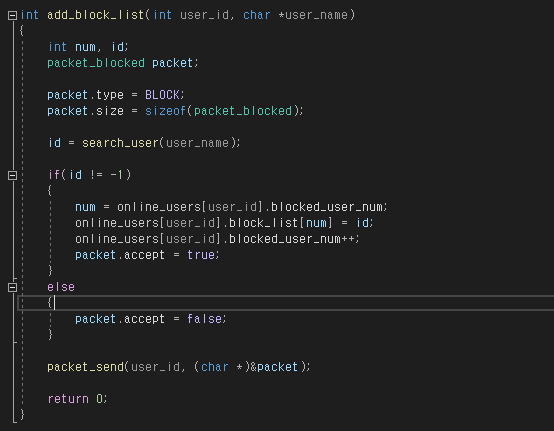
#### Find\_empty\_room(완)



방 배열의 처음부터 탐색해 비어있는 곳의 index를 반환합니다.

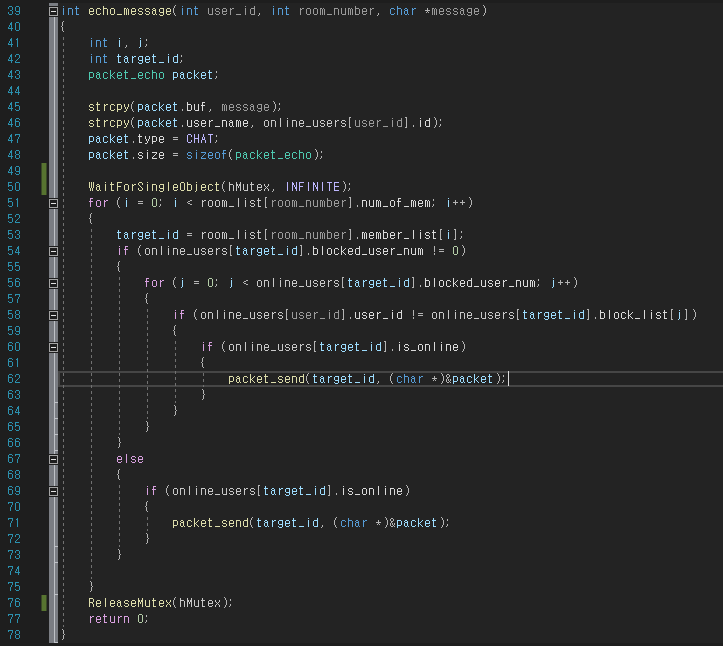
### Chat\_processor.c(완)

#### Add\_block\_list(완)



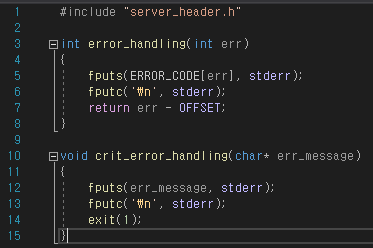
User\_name을 받아 그 존재 여부를 확인 후, 해당 유저를 차단 목록에 추가하고 결과 패킷을 전송합니다.

#### Echo\_message(완)



Id, room\_number, message를 받아 해당 방에 접속한 유저 전체에 메시지 패킷을 전송해 유저가 보낸 채팅을 볼 수 있게 합니다. 차단 목록에 있는 유저에겐 패킷을 안 보내 채팅을 볼 수 없게 합니다.

### Error\_handling.c(완)

error\_handling함수는 전달받은 오류코드에 해당하는 오류 메시지를 출력합니다. 주로 패킷과 관련된 오류메시지를 출력할 때 사용합니다.

Crit\_error\_handling함수는 전달받은 오류 메시지 자체를 출력합니다.

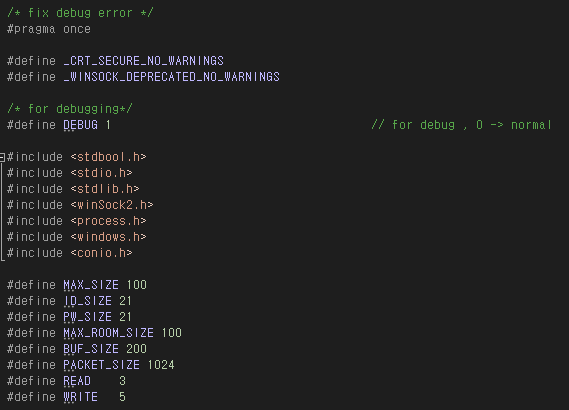
주로 소켓 연결과정에서 발생하는 오류메시지를 출력할 때 사용합니다.

# Client(완)

## **헤더파일(완)**

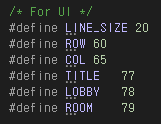
### Client\_header.h(완)

#### 전처리(완)



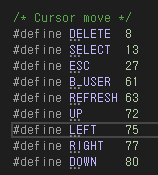
Server\_header와 대부분 유사하지만 콘솔 입출력을 위한 conio.h 가 추가되었습니다.

#### UI (완)

UI에서 사용할 전처리 지시자입니다.

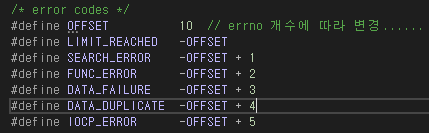
콘솔의 크기 및 각 화면전환에 필요한 값을 지정했습니다..

#### Cursor(완)

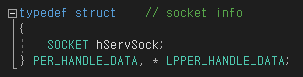
키보드 입력으로 커서를 움직이기 위해 필요한 입력별로

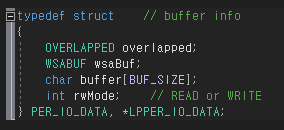
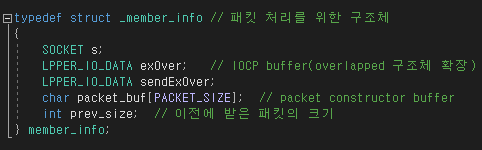
값을 지정했습니다.

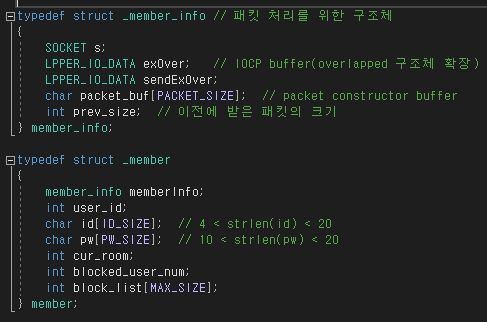
#### 오류코드(완)

서버의 그것과 동일합니다.

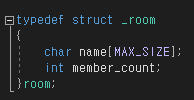
#### 데이터 구조체(완)

 server에서와 다르게 별도의 주소는 필요하지 않기에 소켓만 들어있습니다.



위 3개 구조체는 서버에서 정의했던 내용을 그대로 사용하고 있습니다.

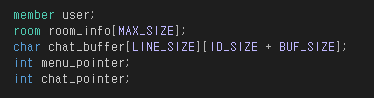
서버와 다르게 member\_list는 제외하였고 멤버 수만 있습니다.

(request 요청시 방의 유저수를 보이기 위함입니다.)

#### 오류코드 메시지 배열(완)

서버의 그것과 동일합니다.

#### 전역변수(완)

내 정보를 저장할 user,

채팅 메시지 저장을 위한 chat\_buffer,

Menu를 가리키는 menu\_pointer,

화면 상 chat 위치를 가리키는 chat\_pointer,

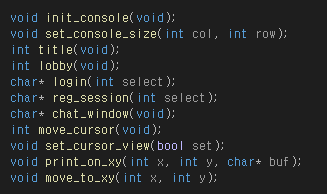
현재 서버에 존재하는 방 정보 room\_info 배열

#### 함수 원형(완)



스레드와 관련된 함수입니다. [main\_thread.c](#_Main_thread.c)에 정의되어 있습니다.

  
패킷 처리 함수입니다. [Packet\_handler.c](#_Packet_handler.c)에 정의되어 있습니다.

콘솔과 관련된 함수입니다. 기본 설정 함수 및

Title, login, lobby, reg, chat 등 각 기능별 콘솔 화면을 담당하는 함수가 있으며 콘솔상에서 입출력 처리를 담당하는 함수가 있습니다.

[UI.c](#_UI.c)에 정의되어 있습니다.



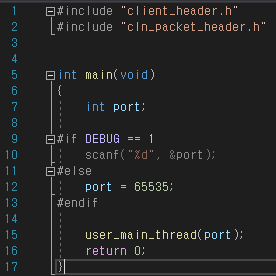
오류처리 함수는 서버의 그것과 동일합니다. [Error\_handling.c](#_Error_handling.c(완)_1)에 정의되어 있습니다.

### Cln\_packet\_header.h(완)

서버에 만들어둔 packet\_header 와 동일합니다. 구분을 위해 이름만 다르게 지정했습니다.

## **소스파일**

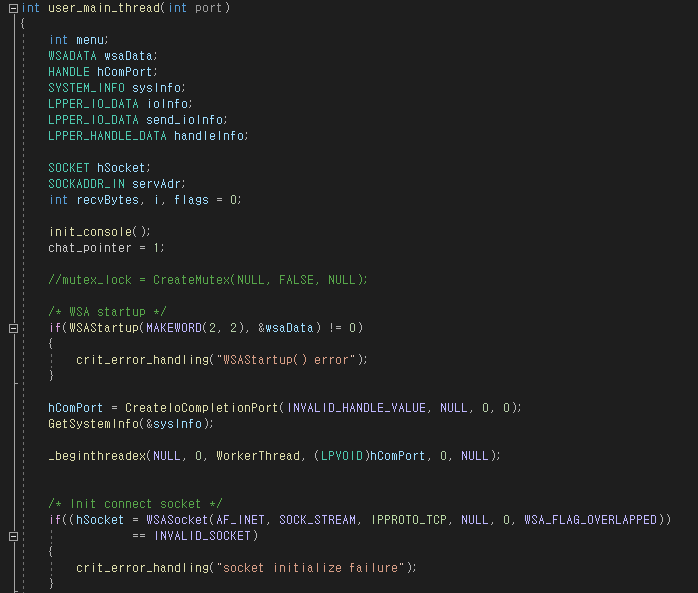
### Main.c(완)

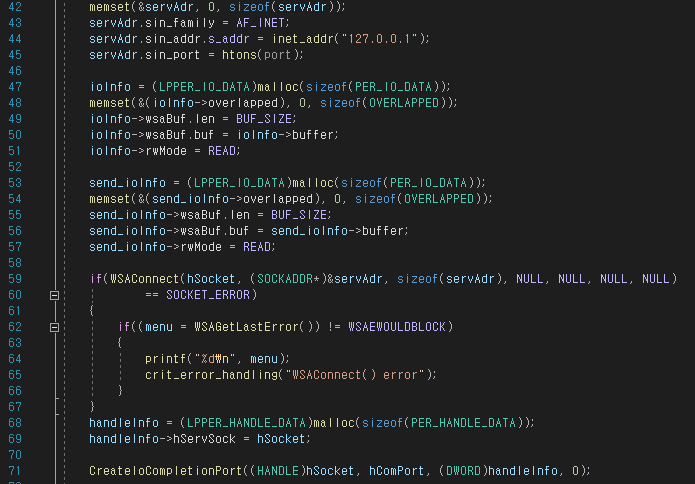
포트번호를 입력받아 main\_thread 함수를 호출합니다.

실질적인 기능은 user\_main\_thread함수 내에서 합니다.

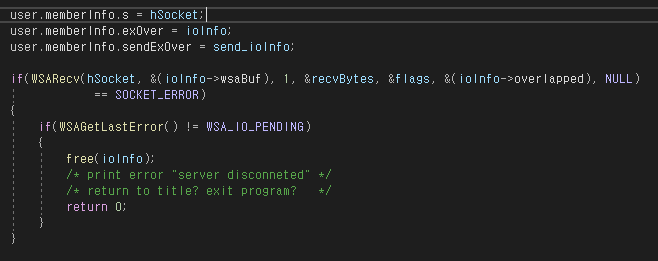
디버그 모드를 만들어 필요시에만 포트번호를 새로 설정할 수 있게 했습니다.

### Main\_thread.c(완)

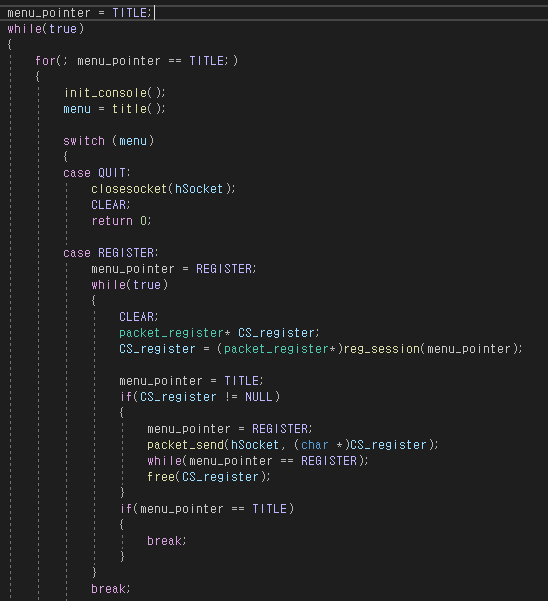




위 두개의 사진은 iocp 모델을 활용하기 위한 소켓의 생성, 서버에 연결, 스레드 생성, CP의 등록 등 기본적인 내용입니다.



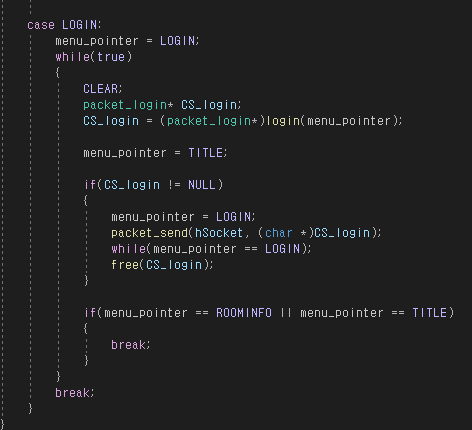
이후 서버의 소켓 정보를 변수에 담아 스레드가 서버로부터 보내는 패킷을 수신할 수 있도록 해줍니다.



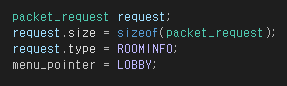
이후 menu\_pointer를 TITLE로 초기화 해주어 title 화면에 진입합니다. 프로그램 실행시 첫 화면은 타이틀이기 때문에 반복문안에 switch 케이스를 두어 타이틀에서 선택한 메뉴에 따라 다른 처리 즉 화면 전환을 위해 이렇게 만들었습니다.

위 사진에서 QUIT은 종료를 선택한 경우입니다.

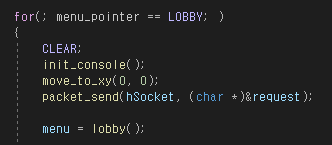
REGISTER를 선택시 reg\_session 함수를 실행해 계정 등록 패킷을 생성, 반환받은 패킷을 서버에 전달합니다.

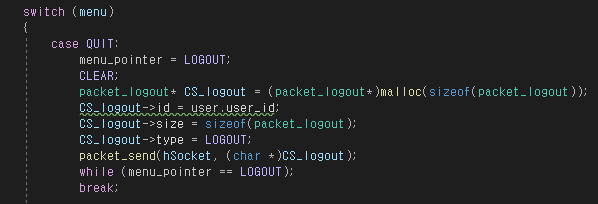


LOGIN 선택 시 포인터를 LOGIN으로 바꾸어 준 후 login 함수를 실행, 반환 받은 패킷을 서버에 전달합니다.

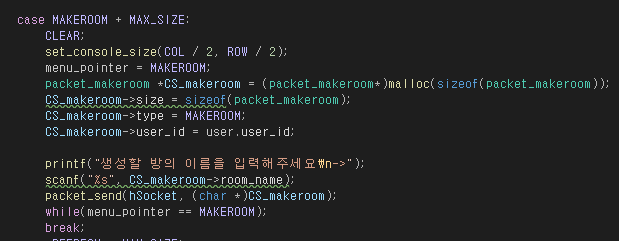
타이틀로 돌아올 때 마다 request 패킷을 만들어 전송해

방의 정보를 갱신합니다.

로그인 성공후엔 LOBBY로 진입합니다.



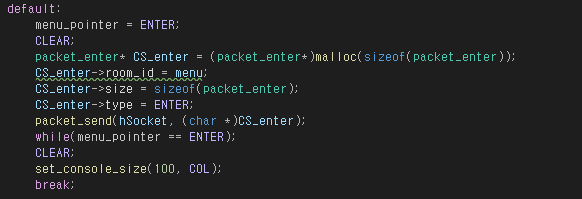
로비에서 ESC입력시 logout 패킷을 보내 종료를 알리고 반복문을 탈출합니다.

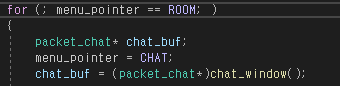


M/m을 입력 시 방의 이름을 받아 방 생성 요청 패킷을 서버에 전달합니다.

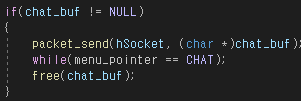


F5키를 눌러 방 정보를 갱신합니다. 반복문이 수행되기 때문에 위에서 봤었던 request 패킷을 다시 전송합니다.

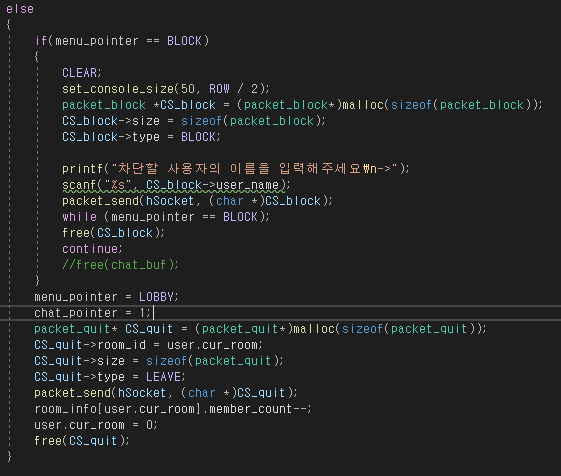




방에 입장 시 실행될 코드입니다. Chat\_window 함수를 실행합니다.

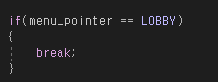


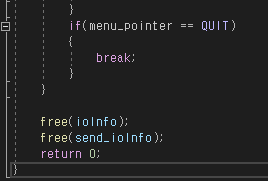
NULL이 아니면, 즉 메시지를 입력했다면 만들어진 패킷을 그대로 서버에 보냅니다.



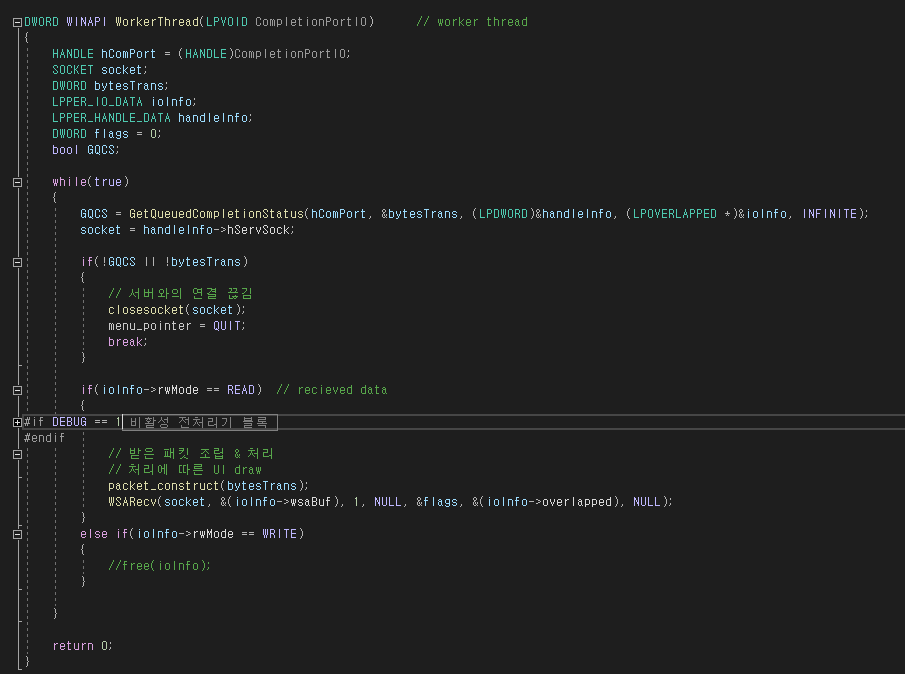
F5를 눌러 사용자 차단 메뉴에 들어가면 유저의 이름을 입력 받아 서버에 차단 목록 추가를 요청하는 패킷을 보냅니다.

BLOCK이 아닌 경우 즉 방에서 나가는 경우엔 방의 멤버 수를 줄이고 서버에 패킷을 보냅니다.

포인터가 LOBBY면 아무것도 수행하지 않고 반복문을 다시 수행해 로비로 진입합니다.



메뉴 포인터가 QUIT이면 반복문을 탈출해 최종적으로 프로그램이 종료됩니다.



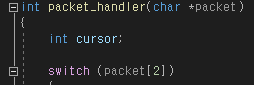
WorkerThread함수는 서버의 그것과 유사하지만 연결이 끊겼을 때 메뉴 포인터를 QUIT으로 바꾸어 서버와 연결이 끊겼음을 인지할 수 있게 했습니다

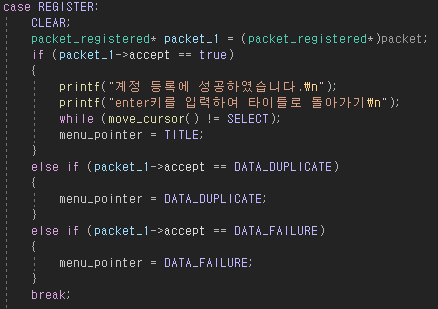
### Packet\_handler.c(완)

#### Packet\_construct(완)

Packet\_construct 함수는 Server의 그것과 동일합니다.

#### Packet\_handler(완)

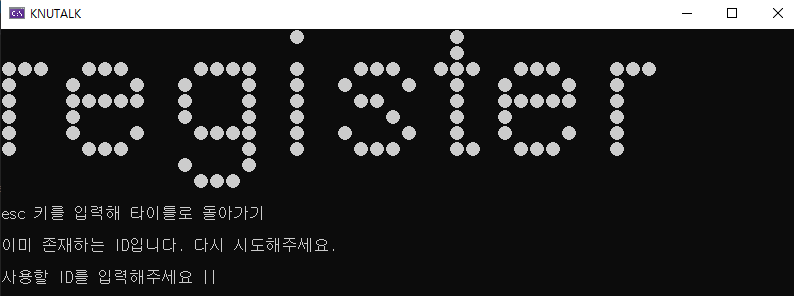
packet\_handler 함수는 packet[2] 즉 패킷의 type을 검사해 각 타입에 맞는 처리를 합니다.



가입 패킷을 받은 경우 성공 여부에 따라 결과 메시지를 출력하고, 엔터 입력 시 타이틀로 돌아갑니다.



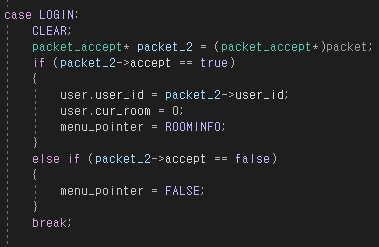
계정 등록 성공시의 콘솔 모습



Id 중복으로 실패한 모습

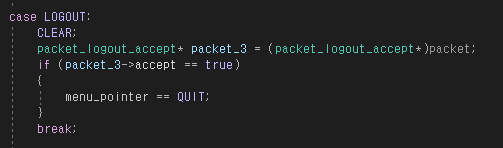


Id, pw 길이 규칙 위반시 모습

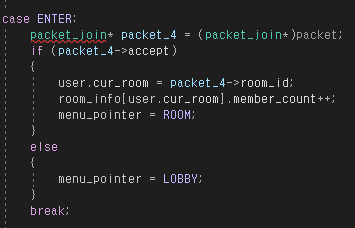


로그인 패킷을 받은 경우, 성공 시 현재 서버내의 정보를 요청

실패 시 오류 메시지를 출력합니다.

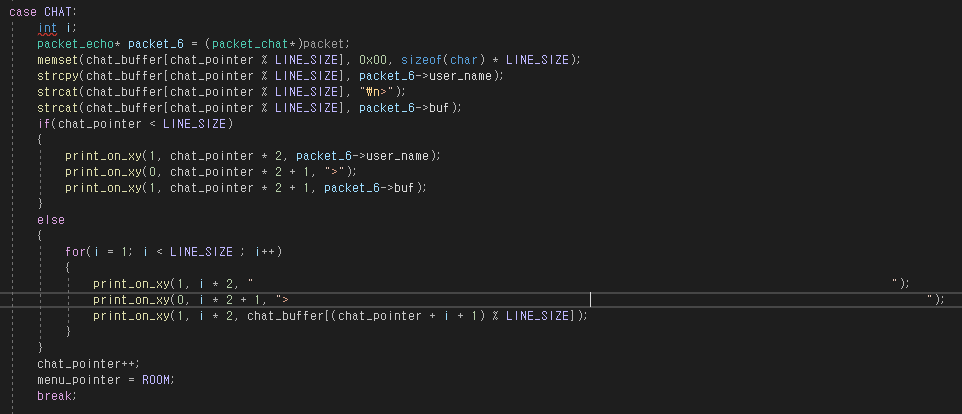


로그아웃 패킷을 받아서 성공여부를 확인 후 menu\_pointer를 QUIT으로 바꿔 종료 상태로 돌입합니다.

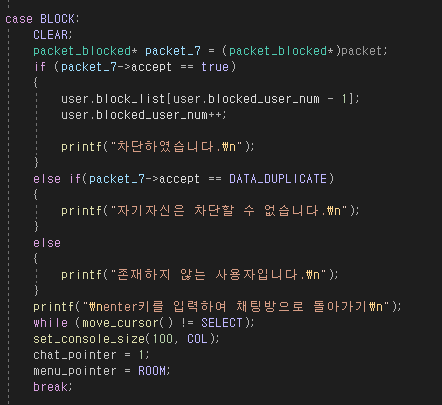


입장 패킷을 받은 경우 유저의 cur\_room을 해당 방의 id로 설정하고 해당 방의 멤버수를 증가시킵니다.

 LEAVE 패킷은 서버에서 이미 echo\_messgage를 통해 처리하기 때문에 별도의 함수를 실행하지 않습니다.

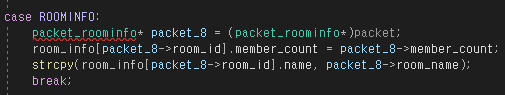


서버가 전송한 에코 메시지를 받은 경우 그 값을 chat\_pointer 위치를 참조해 출력해줍니다.



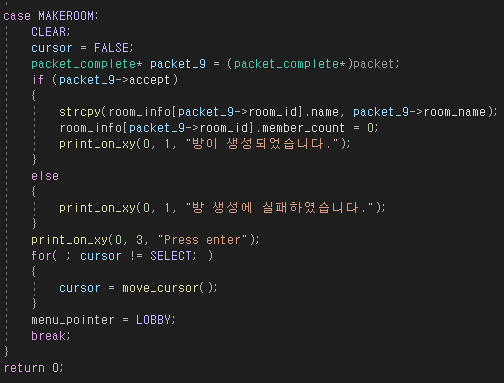
차단 패킷을 받은 경우 성공 시 차단 유저 목록에 추가합니다.

실패 시엔 오류메시지를 출력합니다.



방의 정보를 요청하는 패킷입니다.

서버 측에서 보내준 방의 정보를 client에 있는 room\_info에 저장합니다.

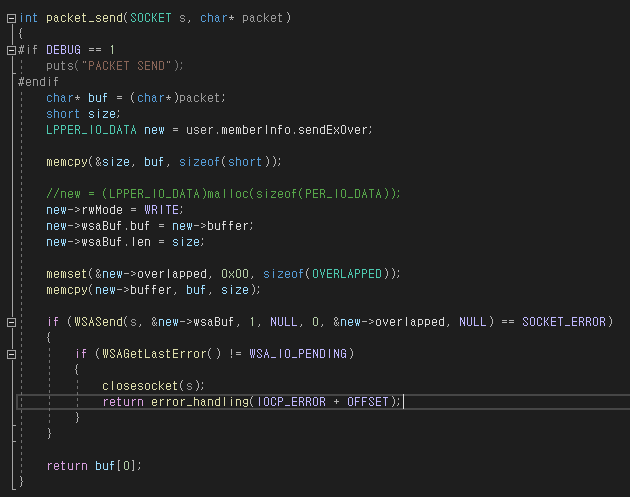


방 생성 패킷입니다. 생성 여부를 확인 후 방 정보를 갱신합니다.

실패할 경우 오류메시지를 출력합니다.

엔터 키 입력 시 로비로 돌아갑니다.

#### Packet\_send(완)



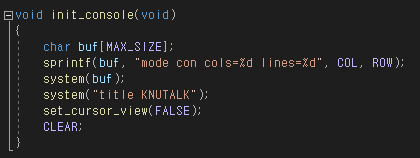
기본적으로 Server에서 정의했던 packet\_send와 유사합니다.

차이점은 Server에선 user\_id를 통해 소켓주소를 획득할 수 있었지만

Client는 그럴 수 없기에 함수 호출 시 소켓 변수를 전달해야 합니다.

### UI.c(완)

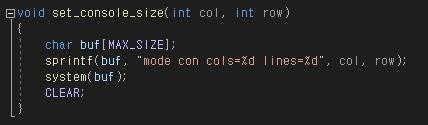
#### Init\_console(완)



콘솔 시작 전, 미리 지정해둔 COL, ROW를 활용해 초기 사이즈를 설정하고 프로그램의 타이틀을 KNUTALK

으로 변경합니다. 커서는 불필요하기에 안 보이게 처리해줍니다.

#### Set\_console\_size(완)



Row, col값을 받아 콘솔창의 크기를 조절하는 함수입니다.

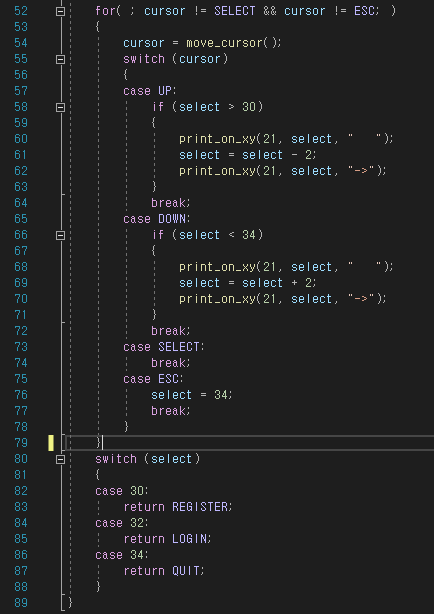
#### Title(완)



Title 함수의 첫 부분으로 콘솔창에 프로그램의 이름인 KNU TALK 및 메뉴를 출력합니다.

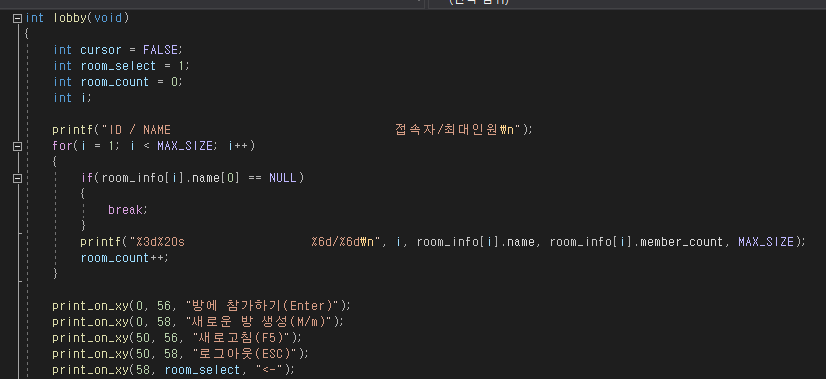


실제 콘솔창에선 위와 같은 모습입니다.

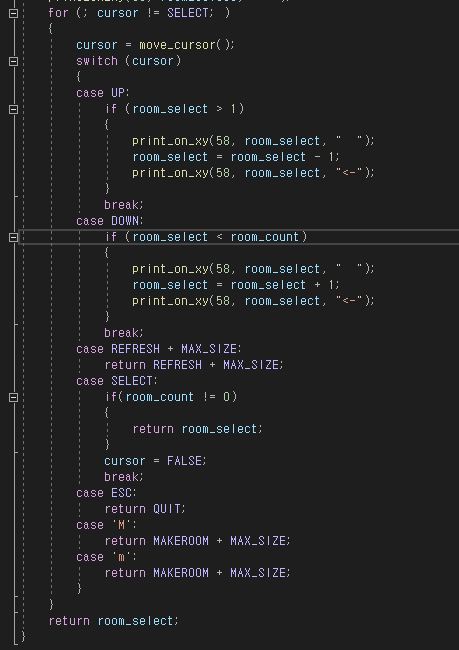


Title 함수의 두번째 부분으로, 화살표키 입력에 따른 커서의 위치를 조절 및 메뉴의 index를 위한 select를 조절합니다. SELECT는 ENTER를 의미하고, 엔터 키를 입력해 메뉴를 선택, 루프를 탈출 후 해당하는 메뉴의 값을 반환합니다.

#### Lobby(완)



Lobby의 첫 부분은 방의 정보 및 각 안내 문구를 출력해줍니다.

로비의 두번째 부분에서는 사용자의 입력을 받습니다.

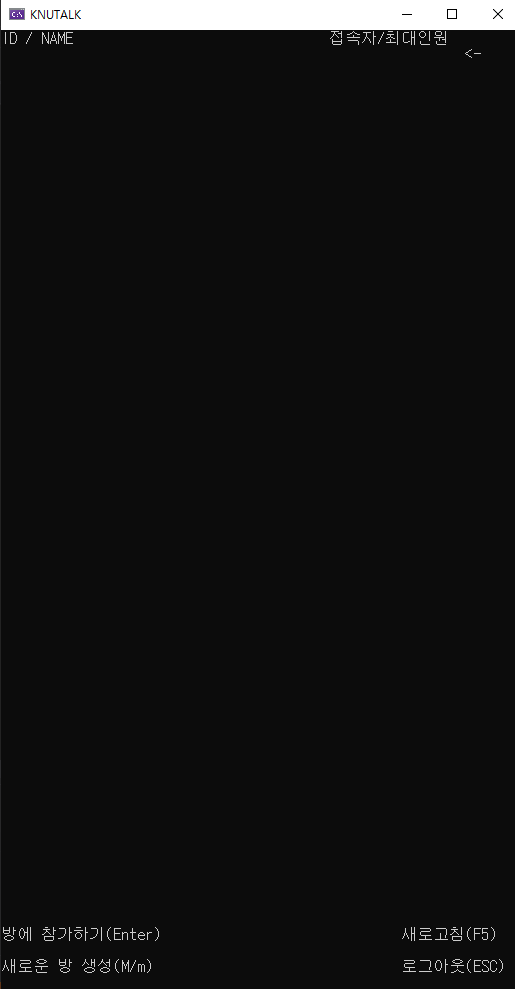
M/m은 방 생성 요청,

화살표 키는 방 선택,

ESC 키는 종료,

F5키는 새로고침입니다.

엔터 키를 입력하면 현재 커서가 가리키고 있는 방에 입장할 수 있습니다.



콘솔 상에서 로비의 모습입니다.

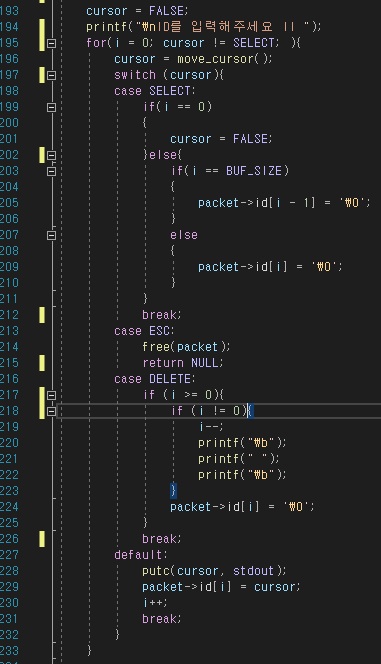
#### Login(완)



Login 함수의 첫 번째 부분으로 로그인 요청 패킷을 만들고, 콘솔상에 login 화면임을 출력해줍니다.

Switch를 통해 전달받은 select의 값을 검사해 로그인 실패 시 오류메시지를 출력해줍니다.

최초 실행시에는 select로 LOGIN 값이 전달되기에 바로 넘어갑니다.

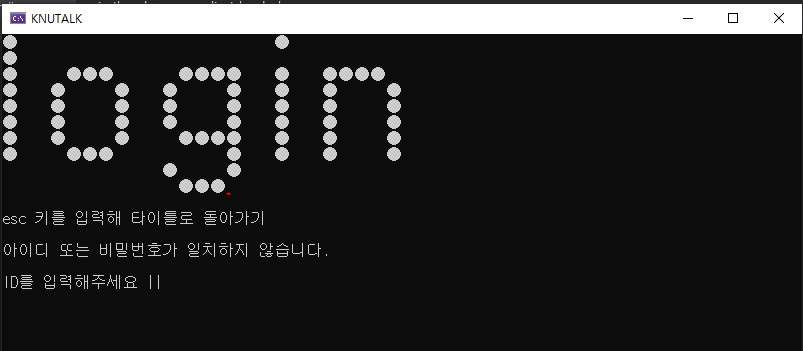


다음은 id를 입력하는 부분입니다. Esc를 누를 경우 타이틀로 돌아갑니다.



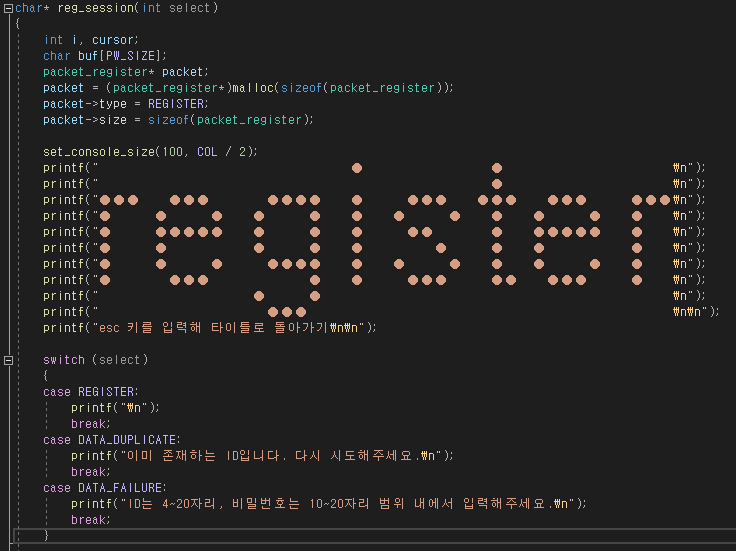
마지막으로 pw를 입력하는 부분입니다. 최종적으로 입력 받은 후 만들어진 패킷의 주소를 반환합니다.

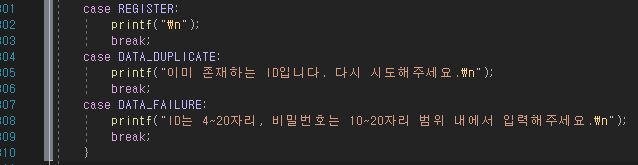
실제 콘솔 상에서의 모습입니다.



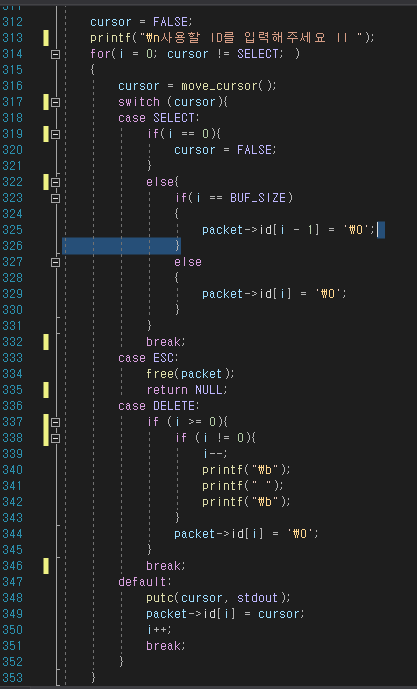
로그인 실패 시 콘솔상에서의 모습입니다.

#### Reg\_session(완)

첫 번째 부분은 계정등록 화면임을 출력해줍니다.



이후 오류 값이 저장되어 있다면 해당하는 오류 메시지를 출력해줍니다.



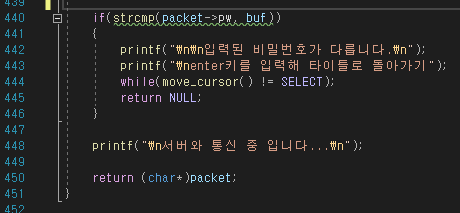
Login 화면에서와 마찬가지로 id를 입력받는 부분입니다.



Pw를 입력 받는 부분입니다.



Login 에서와는 다르게 계정 등록시에는 비밀번호를 두 번 입력해 그 일치 여부를 확인합니다.



두 번 입력한 비밀번호가 일치하면 만들어진 로그인 요청 패킷의 주소를 반환합니다.

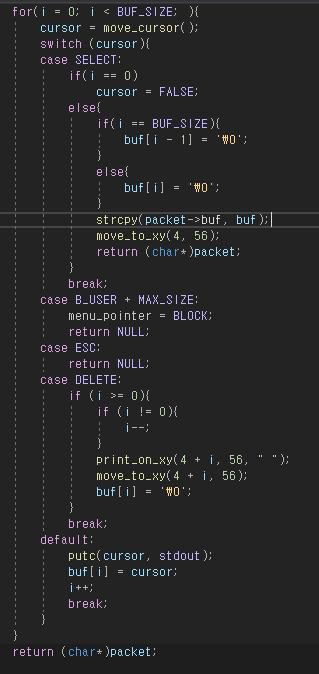


실제 콘솔 상에서의 모습입니다.

#### Chat-window(완)



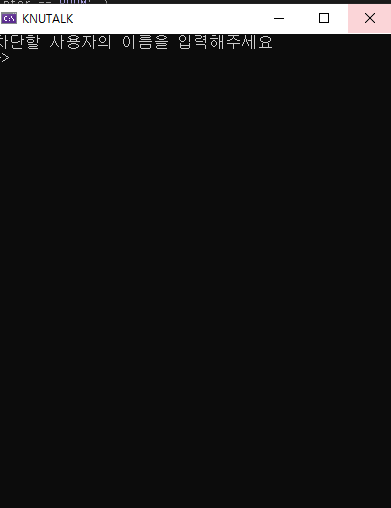
Chat\_window의 첫 부분입니다. 채팅 패킷을 만들기 위해 변수를 만들어 주었고, 메뉴에 대한 안내문구를 출력해준 모습입니다. -> 모양은 메시지 입력창의 위치를 나타냅니다.

채팅 메시지를 받는 부분입니다.

메시지 입력 후 엔터 키를 입력하면 만들어진 패킷의 주소 값을 반환합니다.

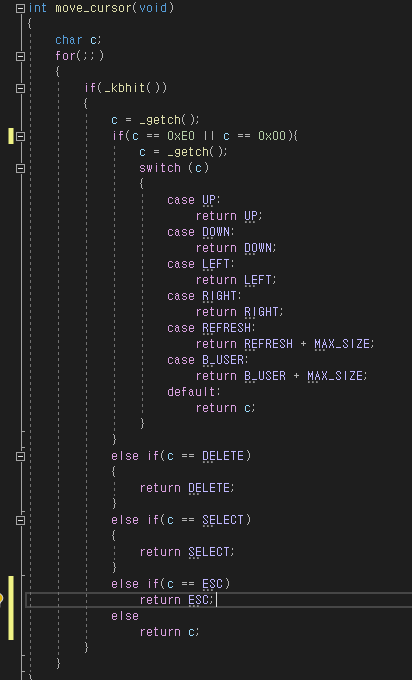
B\_USER + MAX\_SIZE는 F3키를 나타내는데,

입력시 사용자 차단 메뉴로 들어갑니다.



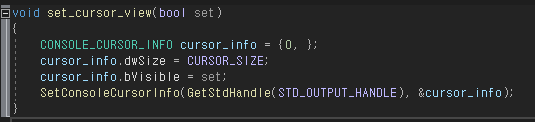
F3 입력시 나타나는 메뉴의 모습입니다. Id를 입력해 차단 목록에 유저를 추가할 수 있습니다.

#### Move\_cursor(완)



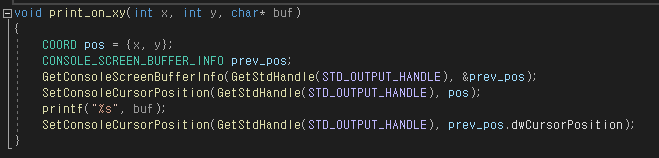
사용자로부터 받은 키보드 입력이 어떤 입력인지 판별한 후 해당하는 값을 반환해주는 함수입니다..

#### Set\_cursor\_view(완)



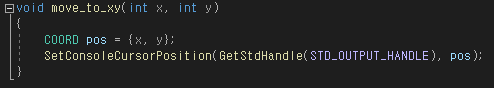
True or false를 값으로 받아 커서를 보이게 할지 안 보이게 할지 변경하는 함수입니다.

#### Print\_on\_xy(완)



콘솔의 좌표 값과 메시지를 받아 화면의 해당 좌표에 메세지를 출력해주는 함수입니다.

#### Move\_to\_xy(완)



좌표 값을 입력 받아 포인터를 해당위치로 이동시키는 함수입니다.

### Error\_handling.c(완)

서버에 만들어둔 파일과 동일합니다.