**샘플 코드 프로젝트**

TCP 기반의 IOCP 채팅 서버

**1. ServerSolution.sln 프로젝트 구성**

**- 작업 환경**

Visual Studio 2015 (v140)

**- LowLib :**

event\_select, iocp, msg 등 핵심 네트워크 라이브러리 로직 및 자료구조, iniReader, Log, Dump 등 시스템에 필요한 유틸 라이브러리 로직으로 구성

**- Server :**

Iocp로직을 이용한 서버 로직이 구현, session 클래스, session manager 클래스 등을 통해 접속된 접속 개체인

세션의 연결 관리를 담당, 프로토콜 처리를 간편하게 할 수 있는 Msg 클래스 및 Session의 Msg 핸들을 통해, 세션 별로 해당 프로토콜을 처리

- TestClient :

EventSelect로직을 이용한 테스트용 클라이언트 구현

MFC 기반 테스트 툴로, 서버와 클라이언트 간의 REQ요청 및 ACK처리의 테스트를 하기 위한 툴

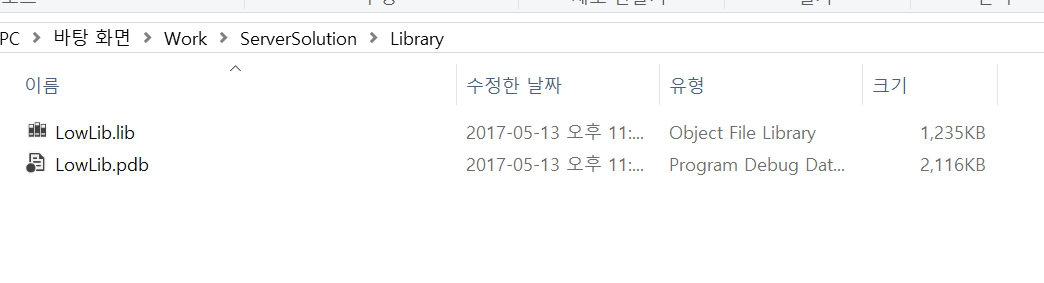
**2. 실행 방법**

1. ServerSolution.sln 빌드

빌드의 결과물들은 각각 별도의 빌드 결과 폴더에 존재

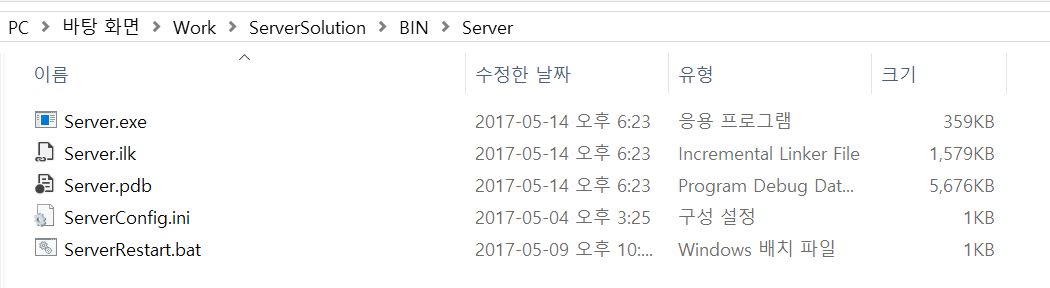
- LowLib :

./Library 폴더



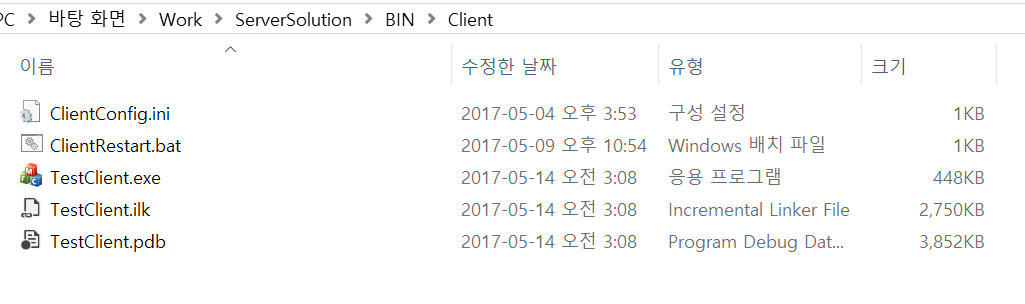
- Server :

./BIN/Server 폴더



- TestClient :

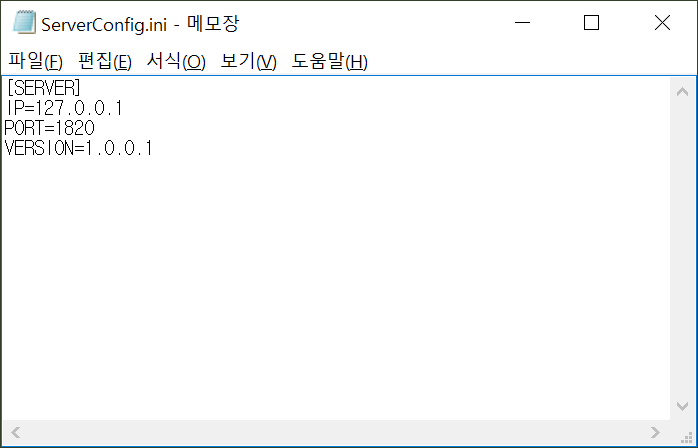
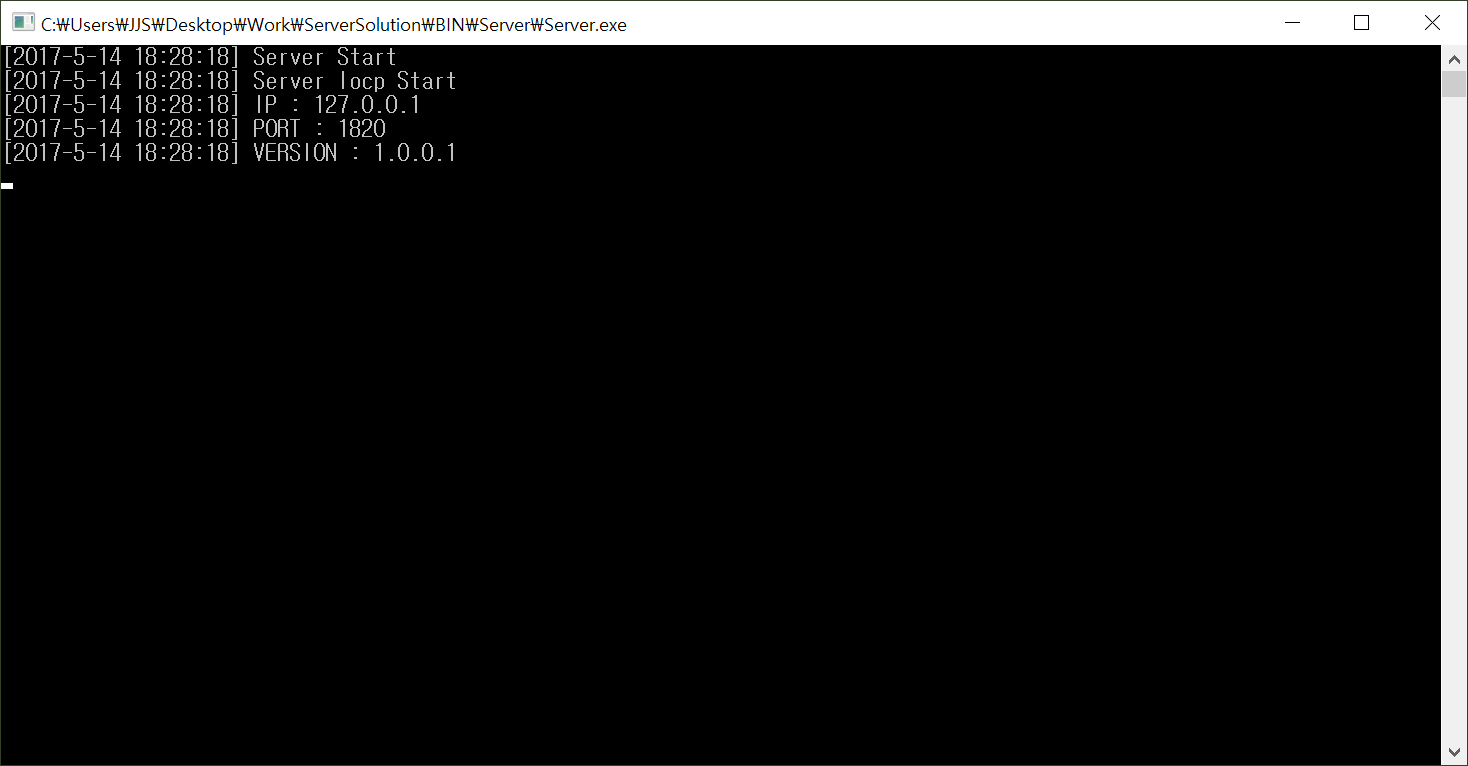
./BIN/Client 폴더



3-1. 서버 프로그램 시작

1) ./BIN/Server 폴더에 ServerConfig.ini 파일이 존재 하는지 확인

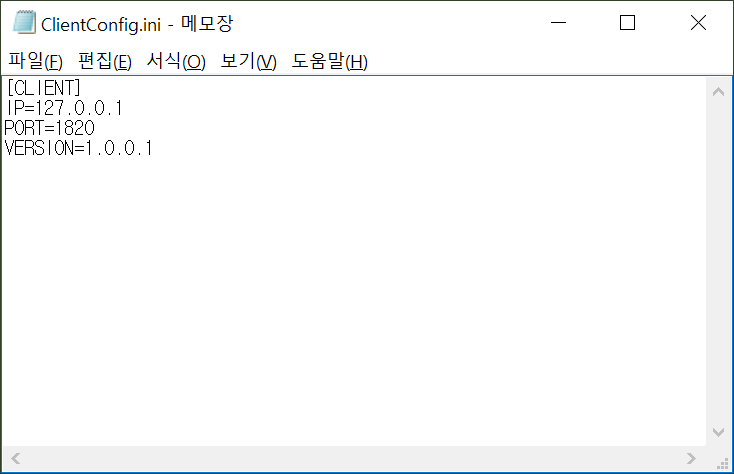
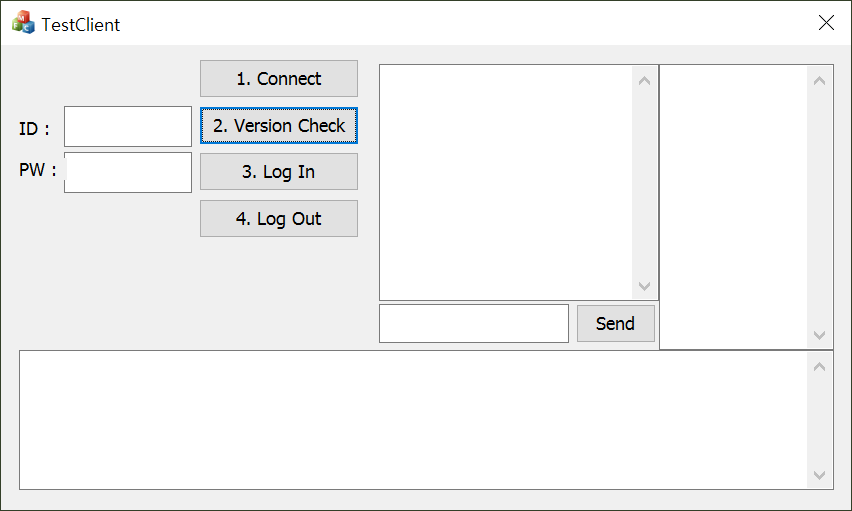
2) ServerRestart.bat 파일을 실행하여 서버를 시작



3-2. 테스트 클라이언트 프로그램 시작

1) ./BIN/Client 폴더에 ClientConfig.ini 파일이 존재하는지 확인

2) ClientRestart.bat 파일을 실행하여 클라이언트를 시작



4. TestClient의 기능

1)Connect : 클라이언트 통신 개체의 초기화 및 서버와의 Connect를 시도

2)Version Check : 클라이언트 버전의 유효성을 검증

3)LogIn : ID, PW를 받아 티켓을 생성, 티켓을 통해 인증이 완료되면 Session의 연결 등록 및 유저 포인터 생성, user\_id 발급 받는 기능

- 참고 : 한번 발급 받은 user\_id는 서버가 떠 있는 한(DB가 연동 되어 있지 않아 있기 때문) 유지 되며 동일한 ID로 접속 하는 경우, 이전의 user\_id를 그대로 발급 받는다.

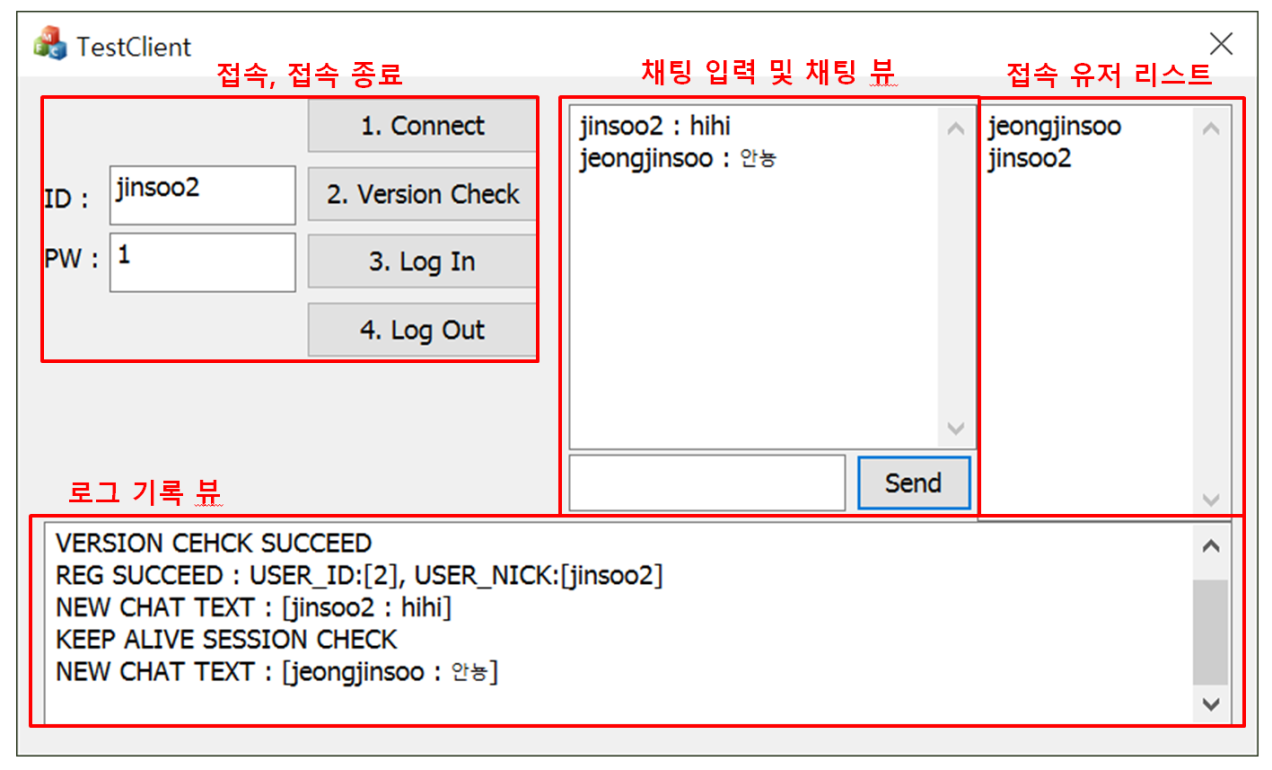
4)LogOut : 유저의 접속을 해제하는 기능

- 참고 : 세션의 연결은 끊지 않으므로, 재 접속 시에 다시 3. LogIn을 통해 접속을 시도할 수 있다

5) 접속 유저 리스트 : 현재 접속된 유저들의 리스트를 출력 (아래 이미지의 접속 인원은 2)

6) 채팅 입력 및 채팅 뷰 : 입력 및 채팅 기록을 볼 수 있는 뷰

7) 로그 기록 뷰 : 클라이언트에서 이루어진 행동의 로그들을 출력하는 뷰



**3. LowLib 프로젝트**

1) Network

|  |  |
| --- | --- |
| **클래스 이름** | **역할** |
| EventSelect | - 소켓의 IO관리를 담당하는 클래스  - 싱글 스레드 기반의 소켓 이벤트를 검사하는 구조  - 클라이언트에서 소켓 처리를 담당 |
| Iocp | - 소켓의 IO관리를 담당하는 클래스  - 다중 스레드 기반, IO의 발생을 담당 스레드가 처리  - 서버에서 소켓 처리를 담당 |
| Msg | - 전송될 버퍼에 필요한 데이터를 담고 조작하는 클래스  - 프로토콜의 ID, 전송될 버퍼로 구성 |
| Network | - 소켓 통신 함수들을 랩핑한 클래스  - 소켓의 등록, 버퍼의 송신, 수신의 기능을 가집니다.  - TCP 기반의 소켓 송수신을 처리 |
| NetworkPacket | - Msg의 송신을 원할 시 Msg 내부에 담겨진 데이터들로 실제 송신에 사용될 패킷으로 변환  - 수신된 패킷을 Msg 형태로 변환하여 반환 |

2) Util

|  |  |
| --- | --- |
| **폴더 이름** | **역할** |
| DataStruct | - 템플릿 기반의 자료 구조를 구현  - Deque, LinkedList, CircleLinkedList, Queue, Stack 로 구성 |
| IniReader | - Ini 파일의 데이터를 추출, 세팅 할 수 있도록 랩핑한 클래스 |
| Log | - 서버에 출력될 로그 기록의 기능  - 서버에 남는 로그를 파일로도 만들어 저장 |
| Memory | - 메모리 관리에 관련된 클래스들이 존재  - AutodetectMemoryLeak : 메모리 누수를 체크하는 클래스, 오픈소스로 외부에서 참조  - MemoryPool : 메모리 new, delete 관리해주는 클래스, 자주 new, delete가 이루어지는 Msg, User의 메모리의 할당 해제를 관리 해 주기 위한 클래스, 오픈소스로 외부에서 참조 및 수정 |
| MiniDump | - UnHandled Exception으로 프로그램이 종료되었을 때, 덤프를 남기는 클래스  - MiniDUmpNormal 사용, 오픈소스로 외부에서 참조 및 수정 |
| Pattern | - 패턴 관련 클래스를 저장  - Singleton : 시스템 내에 단 하나의 클래스 객체를 가질 수 있도록 관리하는 클래스 |
| StringFunc | - std::string, std::wstring 관련 문자열 처리를 담당할 함수들의 모음 |
| Thread | - 스레드 관련 처리를 담당하는 클래스의 모음  - CriticalSection : 임계영역 진입에 사용될 동기화 객체  - ThreadSync : 정의된 동기화 객체를 통해, 스레드의 임계 영역 설정을 제어하는 클래스, 동기화 객체인 CriticalSection의 Lock관 UnLock 사이의 짝이 맞지 않아 생기는 데드락 현상을 예방할 수 있도록 랩핑 |

**4. Server 프로젝트**

1) Content

|  |  |
| --- | --- |
| **클래스 이름** | **역할** |
| User | - 서버에 접속한 유저의 게임 내 정보를 관리하는 클래스 |
| UserManager | - 유저들을 관리하는 클래스, 같은 ID의 유저의 중복 접속을 방지, 접속한 유저를 찾는 등  - 전체 유저를 대상으로 하는 작업을 담당 |

2) Main

|  |  |
| --- | --- |
| **클래스 이름** | **역할** |
| Session | - 접속 개체의 통신을 관리하는 클래스  - Session의 접속 여부, Msg 송수신 요청을 담당 |
| SessionManager | - 접속 개체인 Session을 관리하는 클래스  - 모든 세션에게 Msg를 전송하는 Broadcast 등 전 세션을 대상으로 하는 작업 담당 |
| SessionMsgHandler | - 접속 개체인 Session으로 Msg 가 수신되면, 수신된 Msg의 프로토콜 ID, 버퍼의 데이터를 통해 알맞은 함수를 호출하고 수행할 수 있는 핸들러 |
| ServerIocp | - Session의 접속을 담당하는 Linsten 소켓을 생성하고, 서버에서 사용될 IOCP의 기능을 상속 받은 클래스 |

**5. TestClient 프로젝트**

|  |  |
| --- | --- |
| **클래스 이름** | **역할** |
| ClientSession | - 접속 개체의 통신을 관리하는 클래스  - Msg의 송수신 및 수신 된 Msg의 핸들링을 담당 |
| ClientNet | - EventSelect를 상속 클라이언트의 IO처리를 담당  - ClientSession으로부터 Msg의 송신을 전달 받거나, ClientSession에게 수신된 Msg의 처리를 요청 |
| TestClientDlg | - TestClient 다이얼로그의 동작을 담당  - 텍스트를 로그 뷰에 입력하거나, 신규 접속 유저가 있다면 유저 리스트 뷰에 추가 하거나 하는 등의 UI 처리를 담당 |

**6. CommonFile**

|  |  |
| --- | --- |
| **파일 이름** | **역할** |
| Protocol | - 송수신된 프로토콜의 식별을 위한 ID enum 값을 관리  - REQ, ACK는 짝을 맞추어 존재해야 하며, 뒤에 아무것도 없는 경우 NOTIFY Msg로 간주 |
| PT\_ErrorCode | - Msg를 통해 처리된 작업의 결과를 error\_code 형식으로 전달  ex) PT\_ERROR\_NONE : 작업 성공, PT\_ERROR\_USER\_NOT\_FOUND : 유저를 찾을 수 없음 |
| PT\_Struct | - 수신된 Msg의 버퍼를 구조화 한 결과를 PT\_Format에 저장  - PT\_ReadMsg에 사용될 구조체 |
| PT\_ReadMsg | - 수신된 Msg로 부터 데이터를 구조화된 구조체(PT\_Struct/PT\_Format)에 담아 뽑아내는 함수  - ReadMsg만 있는 이유는 Msg에 데이터를 쓰는 과정은 작업자가 처리하기 쉬우나, Msg를 받아 데이터를 뽑는 과정에서는 어떤 자료형으로, 순서로 뽑는지 잘 알지 못한다. 때문에, ReadMsg에 추가된 함수를 이용하면 쉽게 데이터를 뽑아 다음 처리에 사용할 수 있음 |