|  |  |
| --- | --- |
| 2018/10/23 | 일부 작성시키고 테스트한 라이브러리 코드를 처음 커밋했다.  서버-클라 사이에 테스트용 패킷은 MessagePacket을 주고 받는 것을 테스트 하였다. 잘 동작한다. 패킷 처리시 아직 동시성 문제는 해결하지 않았다. 잘 알려진 Job처리를 넣을 예정. 추후 구글프로토버퍼를 붙일 예정이다.  ServerNetWorkDepartment 가 서버를 작동하는 클래스이다.  - \*\*\*Processor 라고 이름 붙여진 멤버들은 GQCS의 Network-IO 이벤트를 처리 하기 위해 존재한다.  - \*\*\*Processor 중 Send, Recv는 Session 클래스에서 Pointer로 가지고 있다. Send, Recv 는 Session이 처리하도록 고민해봄직 하다. 너무 타고들어가는 느낌이 있다.  ThreadManager  - 생성된 Thread들을 Department 라는 enum을 두어 저장해두고 있다.  - ThreadId는 TLS 로 저장해두고 있다.  - Server 클래스와 , PacketProcess 클래스 에서 따로 Thread Array를 멤버로 가지고 있는데 필요가 없을 듯하다. (추후 삭제하자) |
| 2018/10/28 | - SendBufferQueue로 Concurrent\_Queue를 사용해보기 위해 새로 추가하였다.  - 아직 정확하게 동작할 코드는 작성하지 않았다.  SendBufferQueue 로 Send가 정상 동작한다면 그 후에 구글 프로토버퍼를 붙여볼 계획이다. |
| 2018/10/29 | - ThreadDepartment 클래스를 추가 하였다. 해당 클래스는 생성된 Thread들을 unique\_ptr로 관리한다.  ThreadManager 클래스에서 Thread 생성하기전 Department 를 먼저 생성하고, Thread를 생성하여야 동작한다.  이렇게 나눈 이유는 추후 Thread에 Job을 넘길때 어떤 ThreadDepartment 에 넘길지 선택하기 위함이다. |
| 2018/10/30 | - Session 클래스의 sessionId를 발급하기 위한 IdGenerator 싱글톤 클래스를 추가하였다.  Session이 생성될때 sessionId가 발급되고, 해당 ID 는 서버 프로그램이 종료될때까지 바뀌지 않는다.  이전에는 SessionPool에서 Session을 가져올때 ID를 발급해주었는데 이렇게 되면 Lock을 걸어야해서 최초 서버 구동시 아이디를 모두 발급해버리게 수정 하였다.  - Lock 클래스의 LockSafeScope, LockManager 클래스를 삭제하였다. 단순히 recursive\_mutex로 Lock을 걸었던걸 변경하여 shared\_mutex, shared\_lock, unique\_lock 으로 사용하도록 한다. 매크로 추가. |
| 2018/10/31 | - 기존의 ServerNetWorkDepartment 클래스는 이제 NetworkDepartment를 상속받아 구현하게 되었다.  이는 추후 서버 간의 연결 등 각 NetworkDeapartment를 관리하기 위함이다.  ClientNetworkDepartment도 추가해 두었지만 아직 구현 하진 않았다.  이들을 관리하는 클래스는 NetworkDepartmentManager 클래스이며,  Std::vector<NetworkDepartment\*> 로 관리하고 있다. |
| 2018/11/04 | 에코테스트를 진행하였다. Send를 할 때 SendBufferQueue에 우선 저장을 해 둔후 보내도록 작업 해 두었다. 1-recv, 1-send 를 기본으로 하는데.. 큐에 쌓지 않고 바로바로 보내는 방법도 생각해보자. n-send |
| 2018/11/07 | Job 클래스를 추가하였다. 오브젝트별로 쓰레드에 잡을 던져 처리하는 방식을 사용하기 위해 존재하는 Base Class이며, MemFuncJob 클래스가 이를 상속받고 있다. Job queue로는 Concurrent\_queue를 사용해볼 예정이다. |
| 2018/11/11 | 에코 테스트를 하던 도중 memory 오염이 발생해 찾느라 혼났다.  Recv시 안 옮겨도 될 recvBuf\_ 로우 포인터의 위치를 계속해서 옮겨주고  초기화 하는 코드가 없어 이상한곳에 액세스해 미정의 동작을 보았다.  해당코드를 삭제 후 테스트해보니 정상 동작하였다. |
| 2018/11/21 | Flatbuffer를 추가하여 에코 테스트를 진행하였다. 상당히 잘 동작한다. |
| 2018/11/26 | Spdlog 라이브러리를 사용하여 로그 클래스를 만들었다.  Spdlog::logger 클래스의 몇몇 함수들을 수정하여 파라미터로 const char\* filename, uint32\_t line, 을 추가하였다.  로그를 남길 시 파일명과 파일의 몇번째 라인에 해당하는지를 남기기 위해서 라이브러리를 수정하였다. |
| 2018/11/28 | LocalClock 클래스를 만들었다. Std::chrono 를 사용하였다.  기본적으로 초, 밀리초를 가지고 날짜를 계산한다.  SYSTEMTIME, std::tm, TIMESTAMP\_STRCUT 등으로 변환 해주고,  날짜, 초를 더해 계산도 가능하다.  현재 spdlog 라이브러리를 utf8 도 사용 가능하게 수정을 해 두었는데,  한글이 깨져서 나온다 확인해봐야 할 듯하다.  추후 UTCClock 도 만들어야겠다. LocalClock 과 큰 차이는 없을 듯하다. |

|  |  |
| --- | --- |
| 2018/11/29 | Spdlog 라이브러리 (로그) 데이터 로테이팅 부분 코드를 수정하였다.  기존에는 파일의 용량이 꽉 채워지면 현재 열려있는 파일에 계속 쓰고, 이전 내용이 저장되었다. 또한 로그 파일의 개수를 입력했어야 했는데 그냥 무한으로 파일을 생성하게 바꾸었으며, 파일 이름에 날짜, 시간 정보를 넣어 언제 생성해 작성되었는지 알기 쉽게 바꾸었다. |
| 2018/12/2 | DB를 연결하고 DBQuery를 실행하는데 성공하였다. 앞으로 DBRecord, 행데이터를 얻어와 컬럼을 읽어 값을 받아 올 수 있는 작업을 해야한다. |
| 2018/12/3 | DB 파라미터를 바인드 하는 과정에 c 언어 Type을 입력하는 부분을 직접입력하지 않고 타입을 찾아서 코드로 넣어주는걸 생각해봐야 할 거 같다.  BindParam 함수들이 오버로딩 되어 있는데, 코드 중복을 좀 피해볼 순 없을까? 고민해봐야겠다. wchar\_t 처럼 고정데이터, 가변데이터를 받아와야 할 경우는 함수가 나뉘어야 할거 같긴하다. |
| 2018/12/05 | Tencent rapid json 라이브러리를 이용하여 json을 읽어오는 코드를 추가하였다.  ServerConfig.json 파일을 추가하여 코드에 하드코딩 되어있던 부분들을 서버연결관련 정보를 모두 바꾸었다.  Rapid json을 사용해서 객체들의 정보를 json으로 쉽게 직렬화 할 수 있는 것 같다. |
| 2018/12/06 | Spdlog 의 rotating\_file\_sink 의 rotate 함수를 하나 추가하여 날짜가 하루 지나면 날짜별로 디렉터리를 생성하게 코드를 추가하였다. 00시00분00초 기준이다. |
| 2018/12/11 | SessionPool 을 BaseSessionPool 을 상속받아 구현하게 코드 수정하였다. 이는 추후에 상속관계의 SessionPool을 만들기 위함이고, SessionPool\* 를 파라미터로 받던 함수들도 모두 BaseSessionPool\*로 변경하였으며, Session 생성은 SessionPool 에서 모두 한 후에 ServerNetWrokDepartment, ClientNetWorkDepartment는 SessionPool에서 가져와 연결 수락, 또는 연결 작업만 진행하게 코드를 수정하였다. |
|  |  |
|  |  |