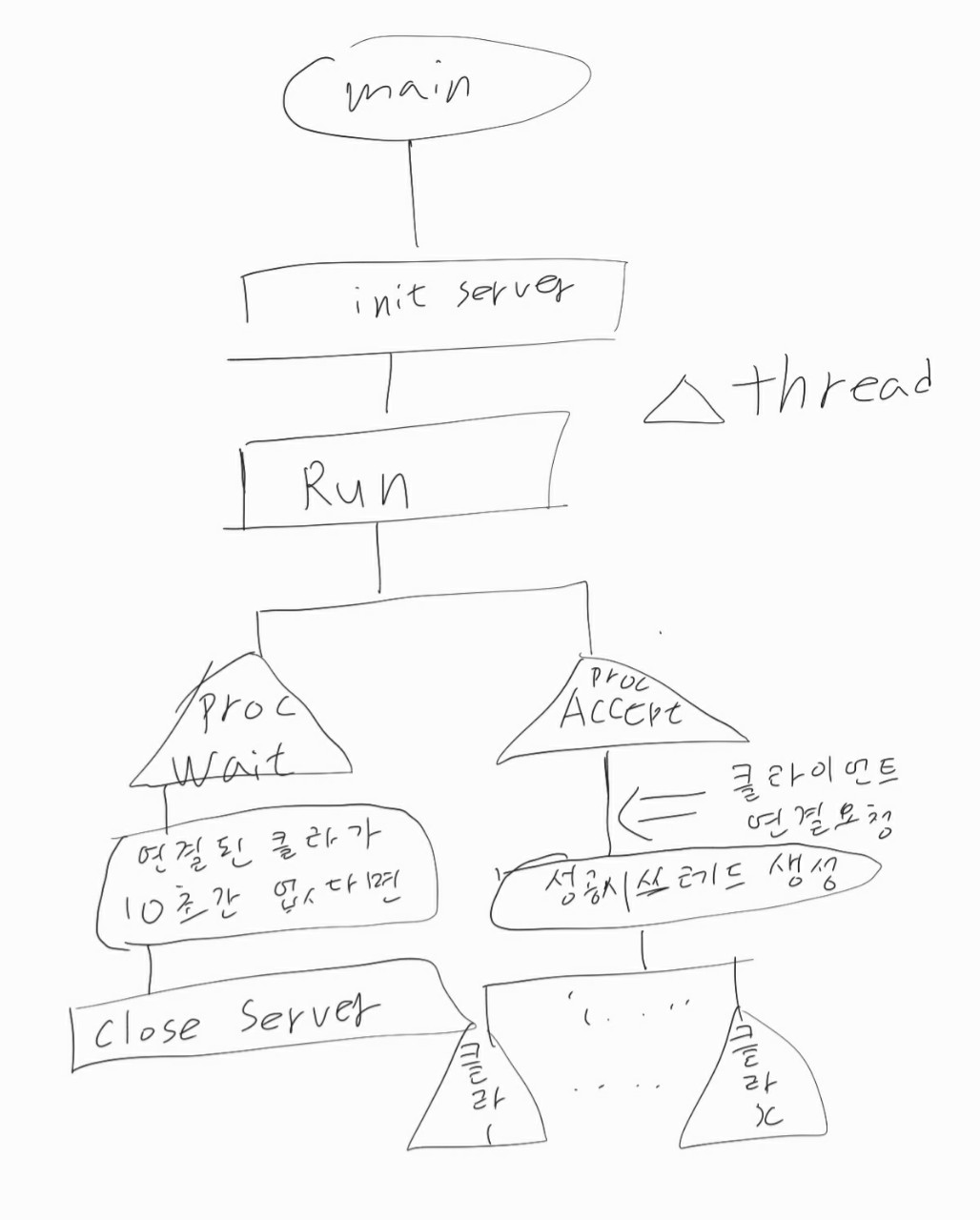
저번 과제의 프로젝트 파일을 그대로 사용해서 서버의 프로젝트 파일 이름이 TCPSingleServer, 클라이언트도 TCPSingleClient로 되있습니다.

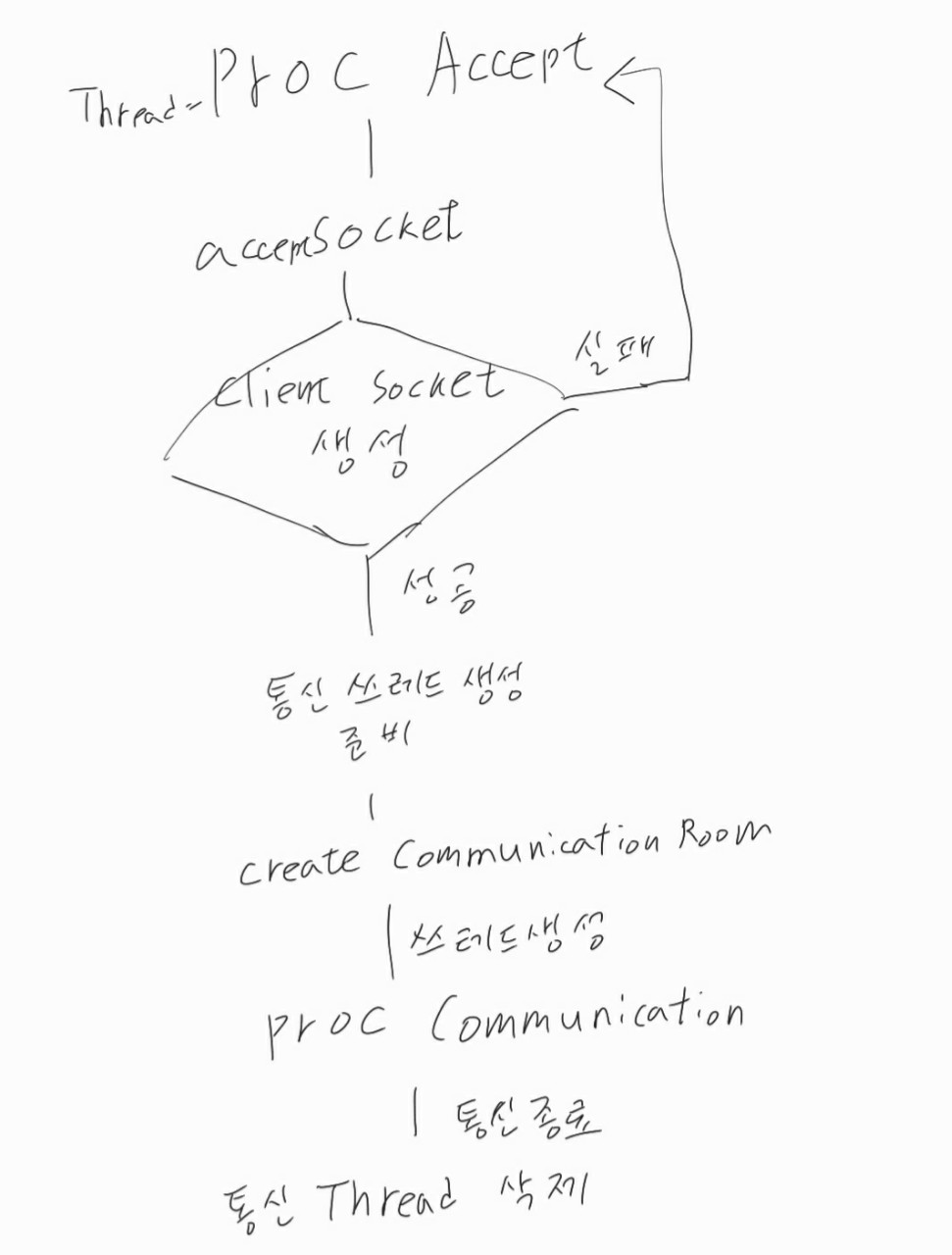
서버 프로젝트에 새로 MultiThreadServer 클래스를 추가해서 main에서 MultiThreadServer 클래스가 구동되게 했습니다.

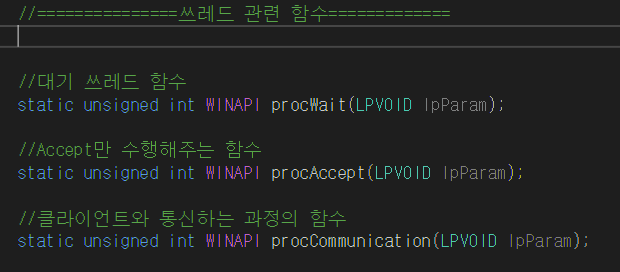
Cpp나 h에 주석을 달아가면서 했지만 코드가 너무 길어져서 새로 만들게 된 것들 중심으로 문서를 만들었습니다.

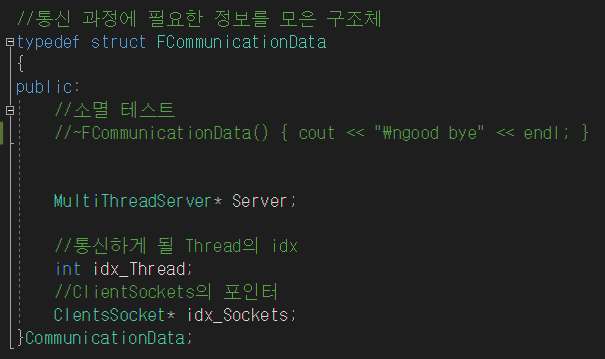
main에서 구동 방식

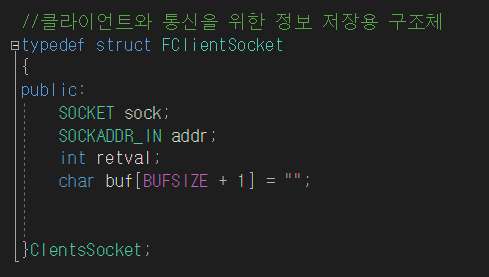
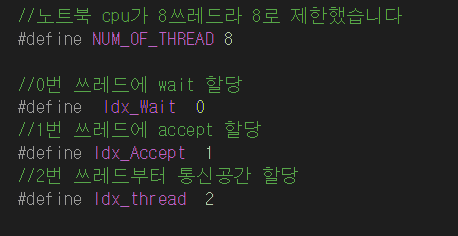
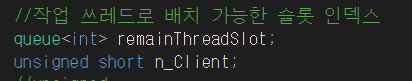


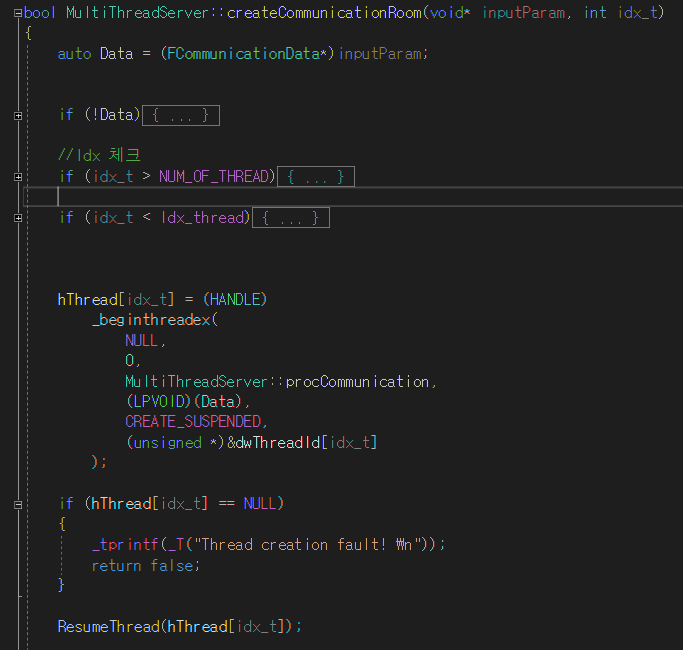
통신 Thread 생성 및 소멸 과정



* 
* Thread 함수들은 MultiThreadServer 내부에 private static 형식으로 선언되어 있습니다.
* procWait은 서버가 일을 하고 있지 않다면 종료시키는 의도로 10초간 클라이언트가 연결되지 않는다면 프로그렘을 종료시키기 위해 bPower를 false로 바꿔줍니다.
* procAccept은 Thread의 남은 자리 수 만큼 클라이언트를 받아주고 Thread를 만들어줍니다.
* procCommunication은 클라이언트와 서버간 서로 데이터를 주고 받는 저번 single server의 통신 코드를 기반으로 구동시켰습니다.



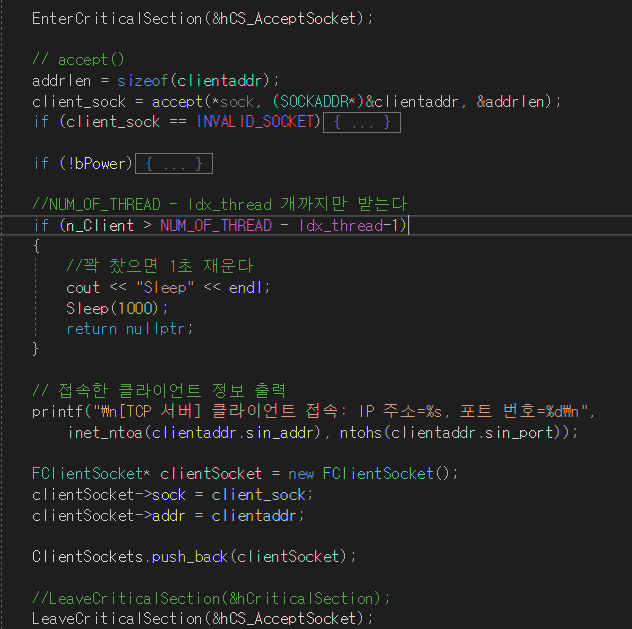
* FCommunicationData는 쓰레드에서 필요한 객체의 정보들을 담아서 입력인자로 보내주기 위해서 만들었습니다. Server는 서버 객체를 가리키고 idx\_Thread는 Thread 인덱스를 저장, 
* idx\_Sockets는 클라이언트 소켓 정보를 가리키고 있습니다.
* 
* 
* 쓰레드 배열을 관리하기 위해서 wait과 accept에 대한 쓰레드의 인덱스를 지정했고   
  그 뒷 번호로 통신 쓰레드가 들어올 수 있게 인덱스를 지정했습니다.
* Queue에 현재 사용 가능한 쓰레드의 인덱스 번호를 넣어두고 지금 접속한 인원 수를 n\_Client에 저장했습니다.
* 생성자에서 2~7이 저장되고 통신 쓰레드가 종료되는 지점에서 그 쓰레드의 인덱스를 remainThreadSlot queue에 push 시켜서 저장하고 다른 클라이언트가 접속을 시도하면 꺼내서 배정시켰습니다.



통신에 필요한 구조체 생성이 완료됐다면 통신쓰레드를 만들기 위해서

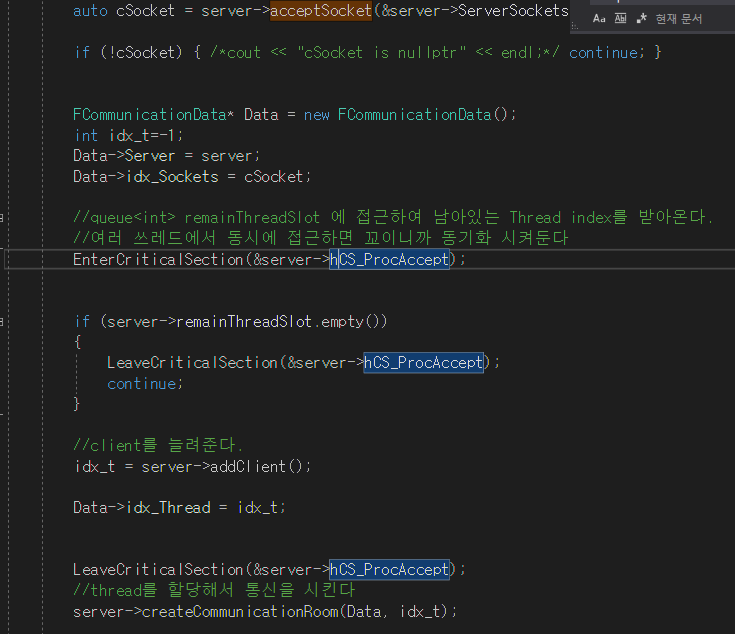
createCommunicationRoom 함수를 호출합니다.

CommunicationData에 대한 검사를 시작하고 이상 없다면 쓰레드를 만들고 작동시킵니다.



FClientSocket\* MultiThreadServer::acceptSocket(SOCKET \* sock) 함수 안에서 CriticalSection을 설정해서 동시에 접속을 시도할 경우 먼저 접속시킨 순서대로 할당될 수 있게 했습니다.

만약 배정된 쓰레드보다 더 많은 클라이언트가 접속을 시도하면 1초간 sleep 시켰습니다.

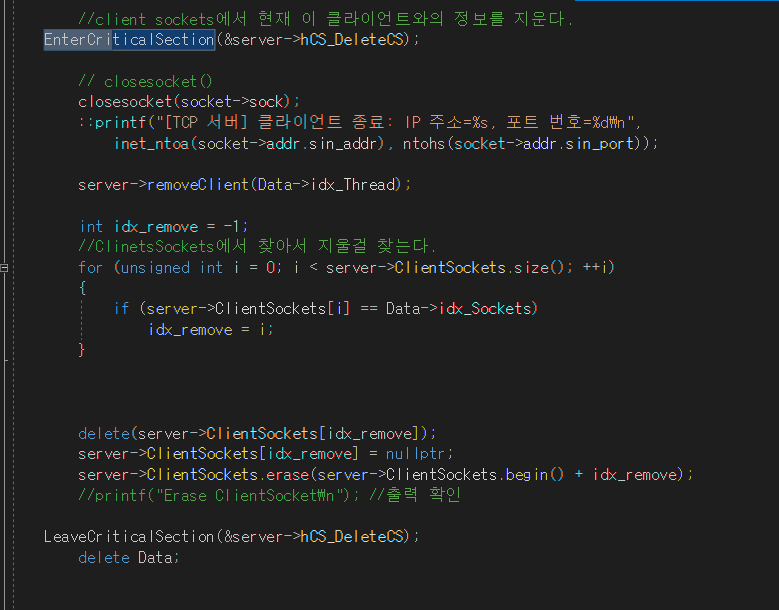


unsigned int \_\_stdcall MultiThreadServer::procAccept(LPVOID lpParam)에서 CriticalSection입니다

addClient에서 n\_Client의 숫자를 늘려줘서 위의 acceptSocket 함수에서 재워주는 조건문에서 -1이 더 붙게 됐습니다.

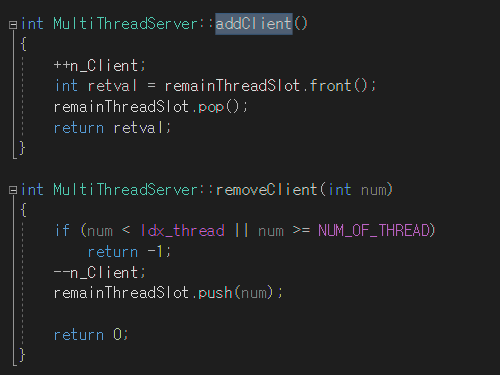
queue에서 남은 자리가 없을 경우 배치를 못 하니 continue로 다시 루프를 돌리게 해줬습니다.

이상없이 통신할 준비가 됐다면 Critical Section을 나와서 통신 쓰레드를 만들게 하였습니다.



unsigned int \_\_stdcall MultiThreadServer::procCommunication(LPVOID lpParam)에서 통신이 끊어지거나 끝나면 해당 소켓을 지우는 과정에서 Critical Section으로 들어가게 하였습니다.

소켓을 닫아주고 클라이언트의 정보를 지우고 할당됐던 Thread의 index 값을 queue에 담아줍니다.



클라이언트가 연결되고 해제될 때 호출해서 조정하는 함수 2개입니다

addClient를 수행하면 리턴 값으로 queue에서 남아있는 Thread idx를 뽑아서 리턴시킵니다.

removeClient에서는 입력인자로 Thread idx를 받아서 queue에 자리가 남았다고 넣어줍니다.