Trial and Error

1. 클라이언트의 입력 도중에 서버로부터 데이터를 받는 상황을 어떻게 처리할 것 인가?

처음에는 main의 while 루프 안에 recv와 입력 부분을 같이 넣어 놨었는데, 이렇게 하니까 입력을 받는 도중에는 recv를 할 수 없어서 원활한 데이터 통신이 되지 않았습니다. 따라서 떠올린 방안이 클라이언트에도 스레드를 넣는 것입니다.

Main의 while 루프는 데이터 입력 및 전송을 하고, 스레드가 recv를 하여 헤더를 분석해 데이터를 받아 화면에 출력하는 것으로 역할을 나눴습니다.

이렇게 하니까 또 발생한 문제가 while 루프에서 인터페이스를 그리는 도중에 recv를 하여 인터페이스 중간이나, 데이터를 입력하려고 대기하는 과정에서 스레드가 출력을 해버려 미관상 좋지 않았습니다. 이를 해결하기 위해 처음에는 Mutex를 이용한 동기화를 했습니다. While 루프에서 인터페이스를 그리기 시작하거나 입력이 감지되면 Mutex를 걸고, 인터페이스 그리기가 끝나거나 입력이 끝나면 Release 했습니다. 스레드에서는 데이터를 출력하기 전에 Mutex를 걸고, 데이터를 출력하고 나면 Release 하도록 했습니다. 이렇게 하니까 또 문제가 발생한 것이, 입력 도중 처음 받은 데이터는 정상적으로 처리되지만, 두번째부터 받은 메시지는 Mutex에 의해서 제대로 처리가 되지 않았습니다.

제가 원하는 결과는 입력 도중에는 서버로부터 받은 메시지가 화면에 출력되지 않고, 입력이 완료되면 여태까지 받은 메시지를 모두 출력하는 것이기 때문에, Mutex나 Critical Section과 같은 동기화 방법은 적합하지 않다고 생각했습니다. 따라서 서버로부터 받은 메시지를 출력하기 전에 출력하려는 문자열을 구조체 배열에 삽입하고, 입력이 종료되거나 입력 중이 아니라면 구조체 배열에 있는 문자열을 for문을 이용하여 순차적으로 출력되도록 했습니다.

2. 연결되어 있는 모든 클라이언트에게 메시지를 어떻게 보낼 것인가?

미리 예제로 주어진 코드에서는 소켓값을 ThreadProc안의 client_sock값에 저장하고 사용하는데, 이렇게 할 경우, 예를 들어 a라는 스레드는 b라는 스레드의 client_sock값에 접근할 수 없습니다. 따라서 떠올린 방안은 결국 소켓값은 int64 자료형이기 때문에, 전역변수로 int형 배열을 선언해서 로그인에 성공하면 배열에 소켓값을 저장하는 방식으로 해결했습니다.

전체 메시지를 send할 때 소켓값에 client_sock_list[i]의 형태로 for 반복문을 사용하여 연결되어 있는 소켓 값에 메시지를 send하도록 했습니다.

또한 서버의 스레드에서는 본인이 가진 소켓값이 client_sock_list의 몇 번째 배열에 존재하는지에 대한 값(sock_number)을 저장하고 있어서, 이후 연결이 해제될 때 client_sock_list[sock_number]를 초기화하여 서버가 연결이 해제된 소켓에는 데이터를 보내지 않도록 했습니다.

3. 이동할 때 id와 소켓값을 어떻게 연관지을 것인가?

식별자로 클라이언트의 id 값을 사용하기 때문에, 서버에서는 각 클라이언트의 id값도 저장하고 있어야 된다고 생각했습니다. 따라서 소켓값을 전역변수로 저장한 것처럼 const char* 자료형의 배열을 선언하여 sock_number의 위치에 id를 저장했습니다. 그리고 이동 명령을 수신했을 때, for 반복문을 사용하여 client_id_list[i]와 수신받은 식별자가 동일할 경우 client_id_list[i]의 소켓값인 client_sock_list[i]에게 이동 명령을 발신하도록 했습니다.