**네트워크프로그래밍 설계과제**



설계과제명: Downloadable Doodle Pad

제 출 일: 2020년 6월 18일

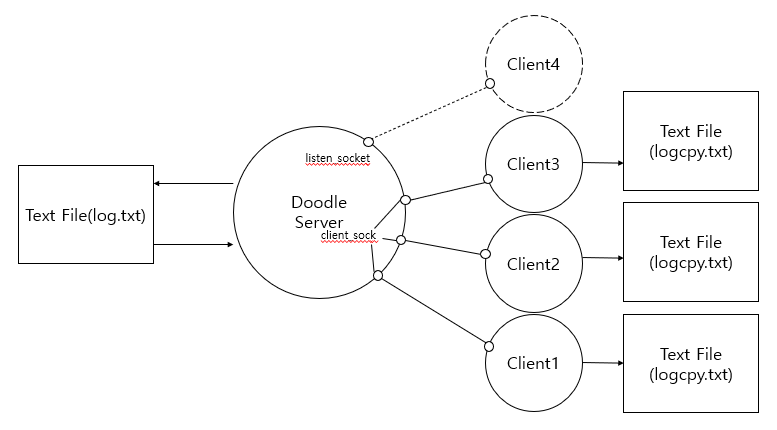
팀 명: Doodling(7조)

팀 원: 2016150019 박재홍

**1. 문제 정의 및 설계과제 목표**

이번 과제를 통해 설계하고자 하는 프로그램은 Downloadable Doodle Pad 즉, 다운로드 가능한 낙서장이다. 기본적으로 클라이언트 측에서 메시지를 전송하면, 서버 측에서 메시지를 받아 파일 I/O를 통해 텍스트 파일에 기록한다. 이것을 소켓 프로그래밍을 이용하여 네트워크 환경에서 다수의 클라이언트가 해당 작업을 할 수 있도록 한다. 거기에 클라이언트 측에서 요청을 보낸다면, 그 파일을 다운로드 받을 수도 있고, 내용을 볼 수도 있다. 다운로드 기능 자체를 구현하는 것은 어렵겠지만 파일 I/O를 통해 복사를 하는 원리를 이용하였다. 결국 같은 내용의 파일을 생성시킨다는 사실은 다운로드를 받는 결과와 같다.

여러 가지 추가적인 아이디어가 있었으나 구현 과정에서 구현하는 데에 한계가 있어 모두 구현하지는 못하였고, 일부 구현된 것도 있으나, 바뀌고 새롭게 추가된 기능도 있다. 핵심 기능(서버 측 파일에 입력, 보기 및 다운로드) 외에 클라이언트에서 서버에 저장된 파일에 덮어쓰는(overwrite) 기능 및 클라이언트 측에 저장된 파일의 내용을 삭제(delete) 기능, 도움말(help)을 출력하는 기능까지 구현하였다. 모든 파일 입출력 기능의 실행 결과는 즉시 직접 파일을 열어 저장된 사항을 확인할 수 있다.

**2. 소프트웨어 설계**

- 메시지

1. 평문을 입력하면 평문을 입력하면 해당 내용이 서버 측 텍스트 파일(log.txt)에 저장된다.

2. log.txt는 !d 명령어로 다운로드가 가능하며, logcpy.txt 파일로 저장된다.

3. 해당 파일의 내용은 !v로 확인할 수 있다. 파일에 저장된 내용이 없는 경우 보낸 메시지를 리턴한다.

4. !o를 통해 서버 측 텍스트 파일에 내용 덮어쓰기가 가능하다. 뒤에 아무 것도 입력하지 않으면 모든 내용을 삭제한다.

5. !x를 통해 클라이언트 측 텍스트 파일인 logcpy.txt 내의 내용을 삭제할 수 있다.

6. !h 또는 !? 명령어를 통해 도움말을 볼 수 있다.

7. !x, !h를 제외한 모든 명령어는 뒤에 문자열을 입력해야 한다. 해당 바이트 수(글자 수)만큼의 데이터를 입출력한다.

8. 평문을 제외한 명령어 입력은 텍스트 파일에 저장되지 않는다.

9. 아무 것도 입력하지 않으면 프로그램을 종료한다.

- 서버

서버는 Windows Socket을 이용하여 구현하였으며, 멀티스레딩 방식으로 구현하였다. 최대로 연결할 수 있는 클라이언트의 수는 지정되지 않았다. 실험 결과 최소 3개의 클라이언트 연결은 무리없이 성공하였다. 서버는 클라이언트의 동작, 즉 전송 메시지에 따라 적절한 결과를 출력하거나, 클라이언트에 다시 메시지를 전달한다.

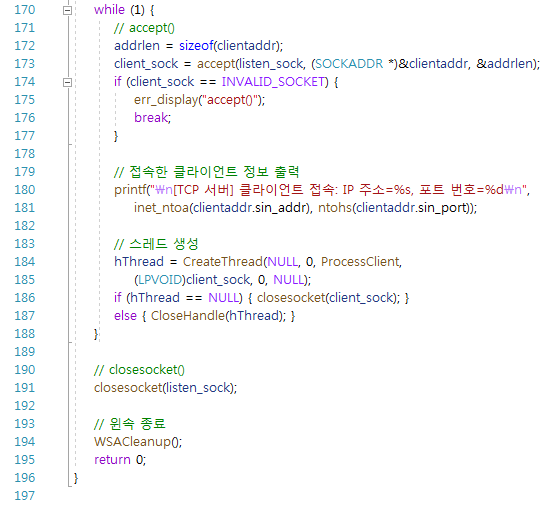
- 클라이언트

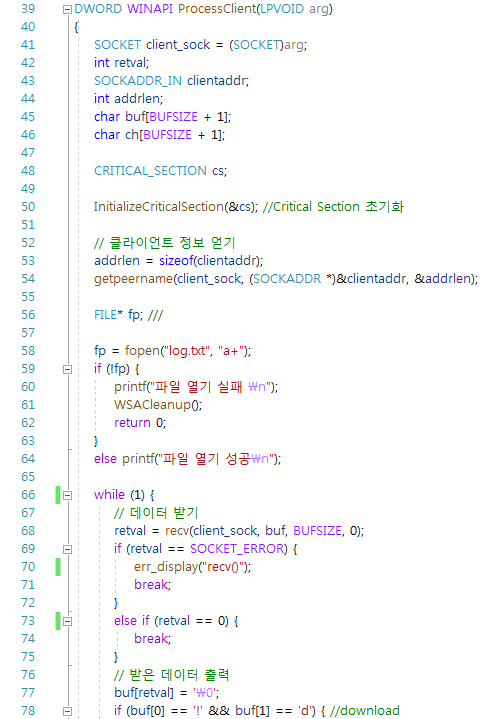
클라이언트도 Windows Socket을 이용하여 구현되었다. 클라이언트는 사용자가 입력한 결과에 따라 다른 동작을 하도록 설계되었다 대표적인 기능으로는 앞에서 서술한 대로 입력, 덮어쓰기, 삭제, 내용 확인, 도움말 보기가 있다. 명령어를 입력할 때의 주의사항은, 도움말 보기와 삭제 기능을 제외한 기능을 이용할 때, 명령어 뒤에 오는 내용이 중요하다. 명령어 뒤에 오는 글자 수(바이트 수)만큼만 입출력을 하기 때문이다.

**3. 소프트웨어 구현 및 데모**

1) 코드 주요 부분 설명

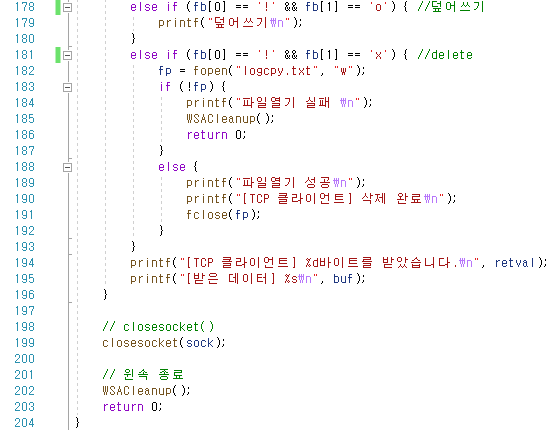
-서버

**** 다음은 서버의 main 함수에서 클라이언트가 접속했을 때 그것을 처리하는 부분이다. 무한 루프로 구성되며, 클라이언트가 접속할 때 스레드를 생성하는 멀티스레딩 방식으로 구성되었다. 연결 가능한 최대 클라이언트 수는 지정하지 않았고, 따라서 그것을 처리하는 부분은 구현되지 않았다.



서버는 평문을 비롯한 다양한 명령어들을 처리하도록 되어 있다. 메시지를 처리하는 동안, 그 중에서 파일 입출력이 있다면 임계 영역을 이용하여 Lock을 걸도록 되어 있고, 입출력이 끝나면 Lock을 해제한다. 파일 입출력은 그때그때 닫았다 여는 동작을 통해 동작 수행 이후 결과를 확인할 수 있다. 이는 클라이언트에서도 동일하다.

- 클라이언트

 클라이언트도 입력된 메시지에 따라 다른 동작이 구현되어 있다. 그러나 입력된 메시지를 서버로 보내는 동작은 기본적으로 동일하며, 서버 측으로부터 반환 받은 메시지에 따라서도 다른 동작을 수행한다. 만약 다운로드와 같은 동작에 관한 메시지를 반환 받는다면, 클라이언트 측의 텍스트 파일에서 입출력이 일어나기도 한다.

2) 데모 시나리오

1. 서버가 실행되고 1번 클라이언트가 서버에 접속한다.

2. 1번 클라이언트가 도움말을 보고(!h), "hello world"메시지를 입력한다.

3. 2번 클라이언트가 서버에 접속하고, 서버 텍스트 파일에 저장된 내용을 확인한다(!v).

4. 2번 클라이언트가 "goodbye" 메시지를 서버 텍스트 파일에 덮어쓴다(!o).

5. 1번 클라이언트가 서버 텍스트 파일을 다운로드 받는다.

6. 3번 클라이언트가 접속하고, "trololo" 메시지를 서버 텍스트 파일에 덮어쓴다.

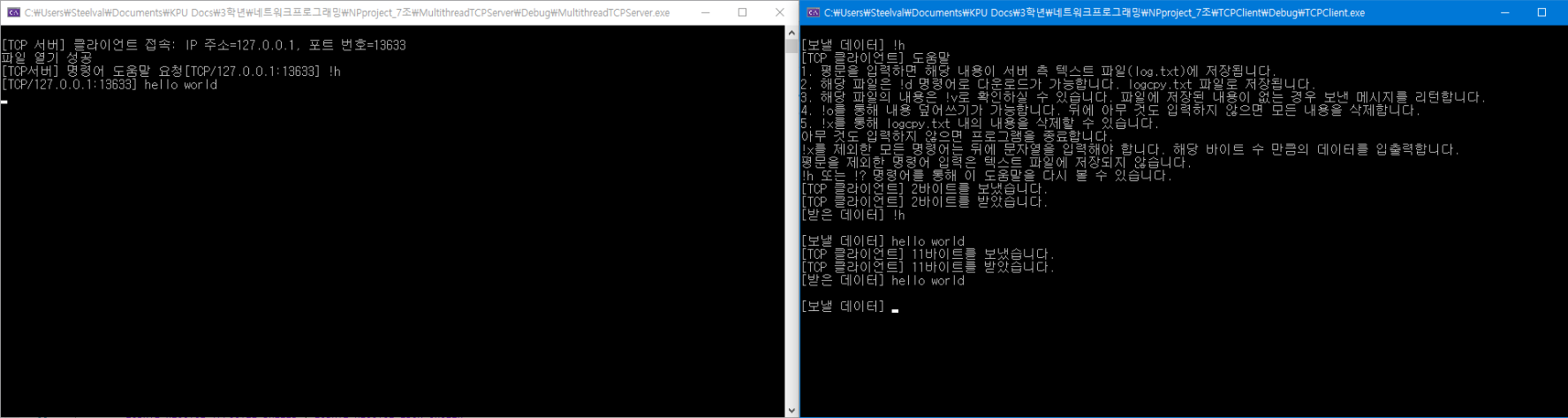
8. 2번 클라이언트가 서버의 텍스트 파일을 다운로드 받는다.

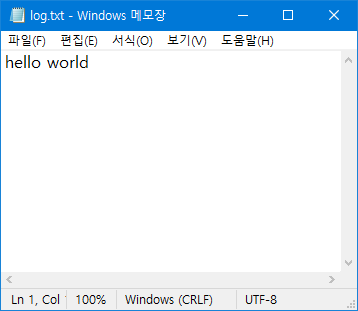
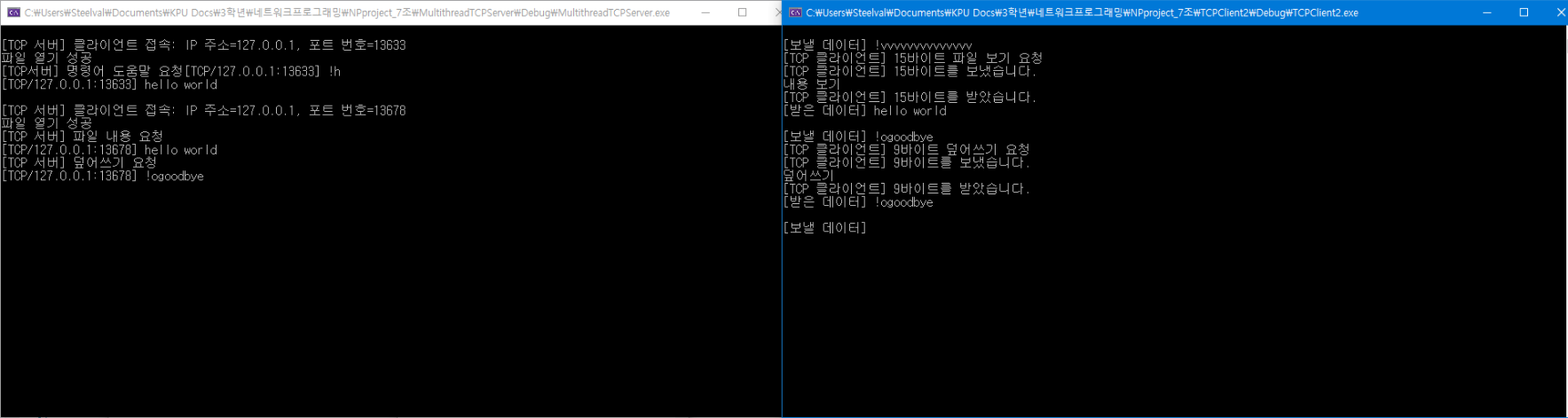
9. 2번 클라이언트가 서버 텍스트 파일에 저장된 내용을 확인하고, 저장된 파일의 내용을 삭제한다.

10. 1번, 2번, 3번 클라이언트의 순서로 모든 클라이언트가 접속을 종료한다.

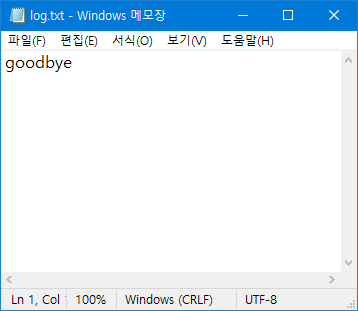
각각의 클라이언트는 동일한 코드를 이용하며, 파일명만 다르다(1번-TCPClient, 2번-TCPClient2, 3번-TCPClient3).

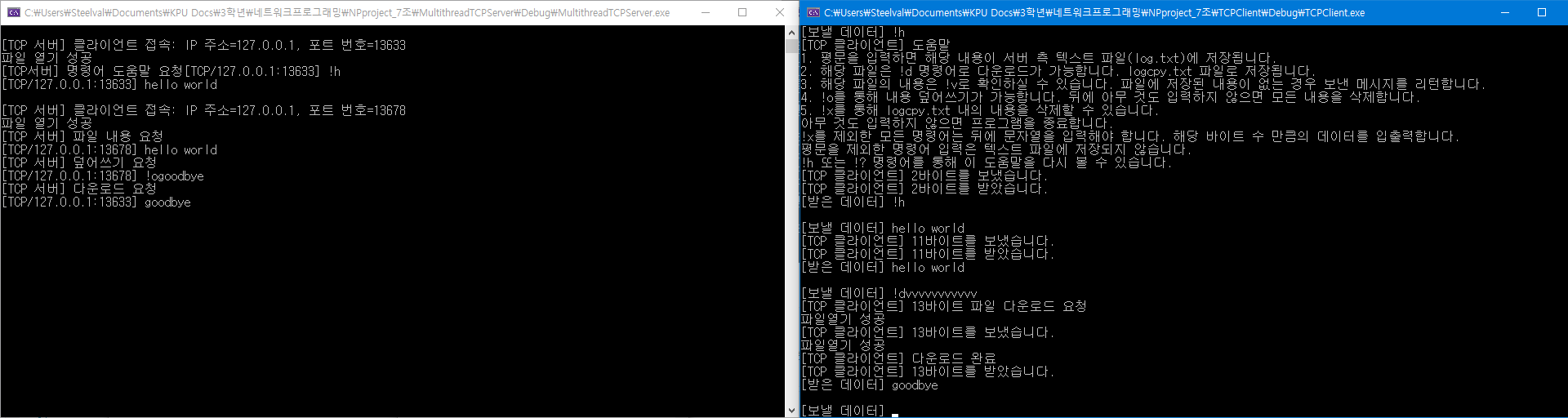
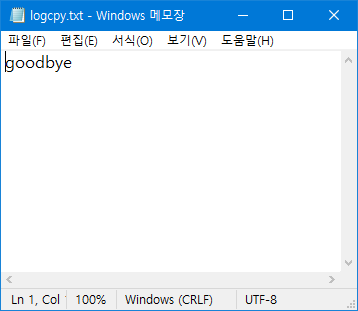
클라이언트에서 log.txt를 다운로드 받으면, logcpy.txt는 각자의 폴더 안에 생성된다.

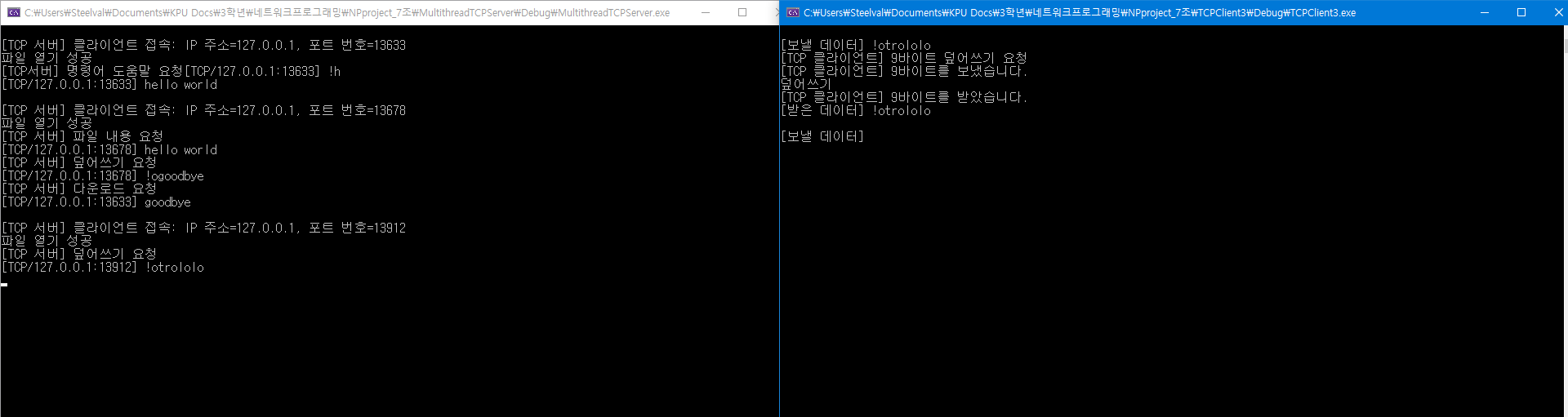
3) 시스템 데모 및 설명(프로그램 실행 결과 캡쳐 및 설명)

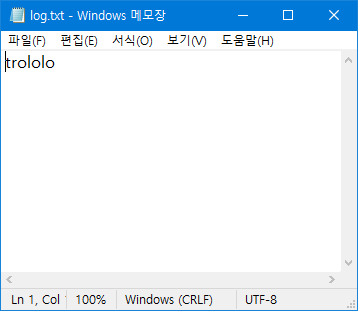
1. 서버가 실행되고 1번 클라이언트가 서버에 접속했으며, 도움말을 출력하고 "hello world"를 입력한 모습이다.

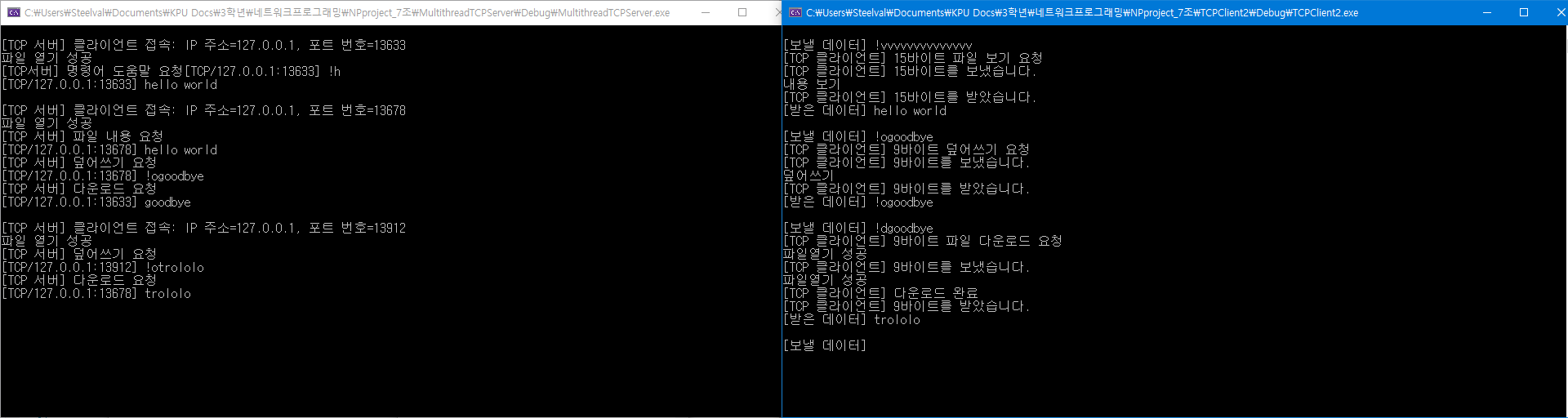
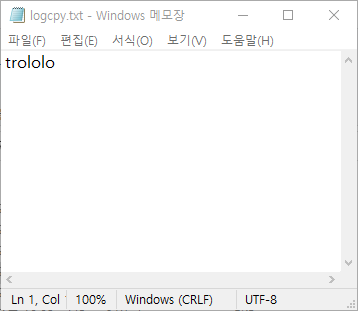
log.txt에 문제없이 메시지가 입력되었다.

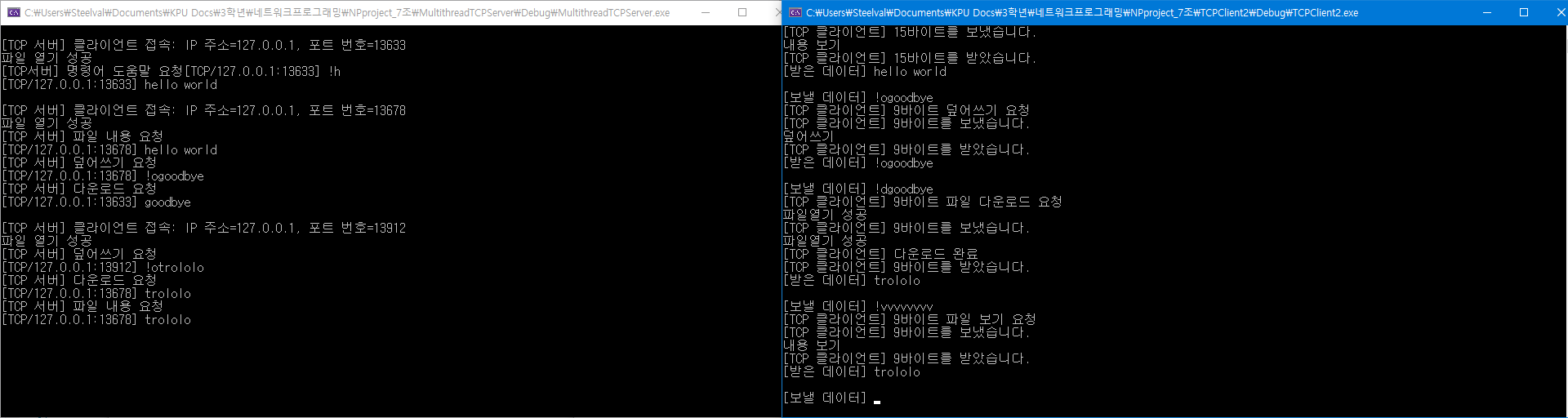
2. 2번 클라이언트가 서버에 접속해서 log.txt 내의 내용을 확인하고, "goodbye" 메시지로 해당 내용 덮어쓰기를 하였다. 이번에도 문제없이 내용이 변경되었다.

3. 1번 클라이언트가 서버 텍스트 파일을 다운로드 받았다.

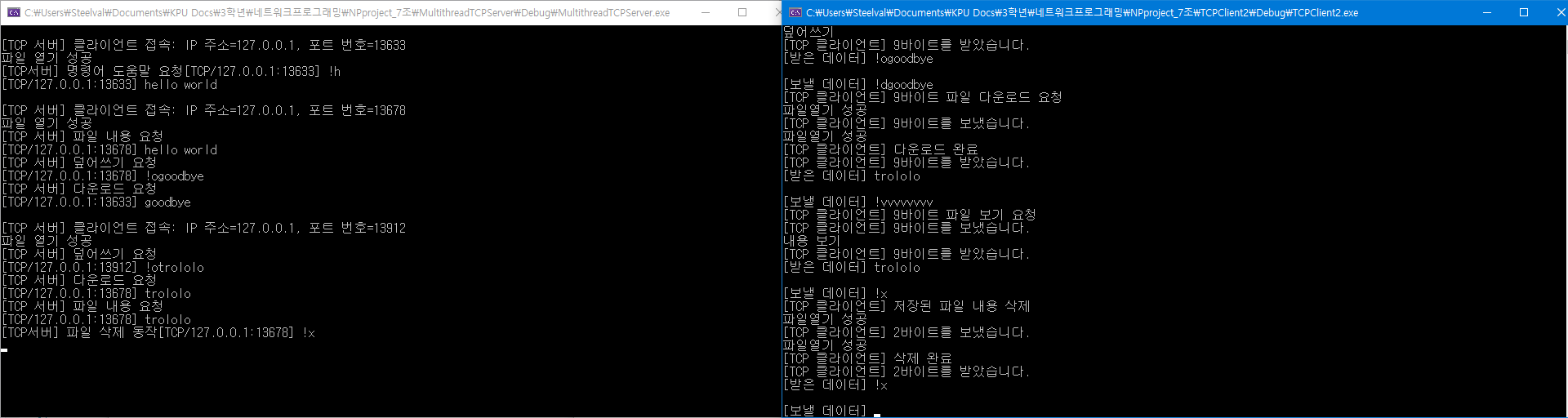
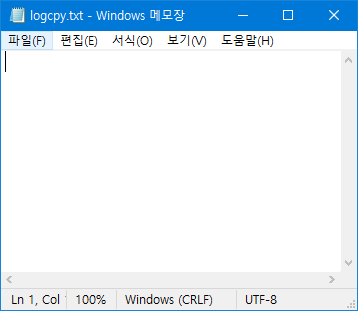
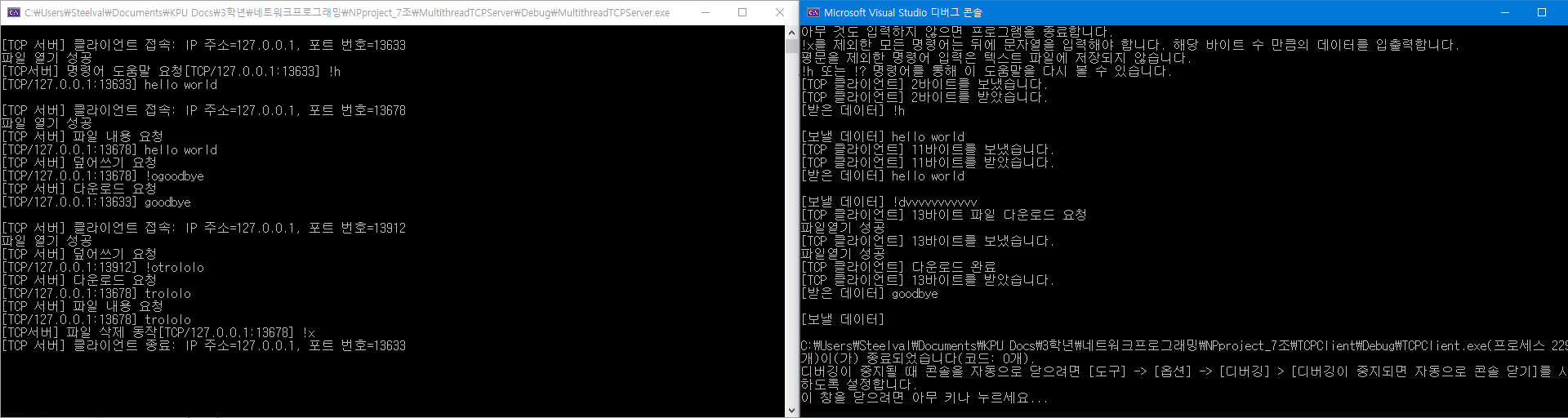
1번 클라이언트 측 logcpy.txt에 무사히 내용이 전달되었다. 다운로드 기능의 구현이다. 3번 클라이언트가 "goodbye"로 덮어쓴 내용이 전달된 모습이다.

4. 3번 클라이언트가 접속한 다음 서버 측 텍스트파일에 "trololo"라는 메시지로 내용 덮어쓰기를 한 모습이다. 이 메시지도 무사히 전달되어 덮어쓰기에 성공하였다.

5. 2번 클라이언트가 다운로드를 요청하였다. 3번 클라이언트가 앞서 내용을 변경하였기에 변경된 내용이 다운로드 되었다.

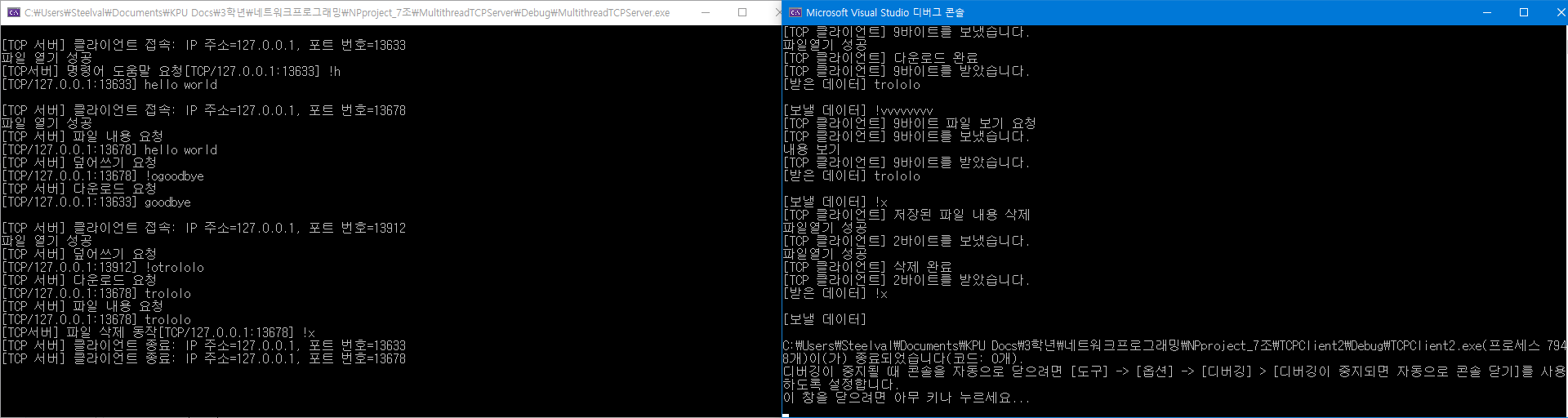
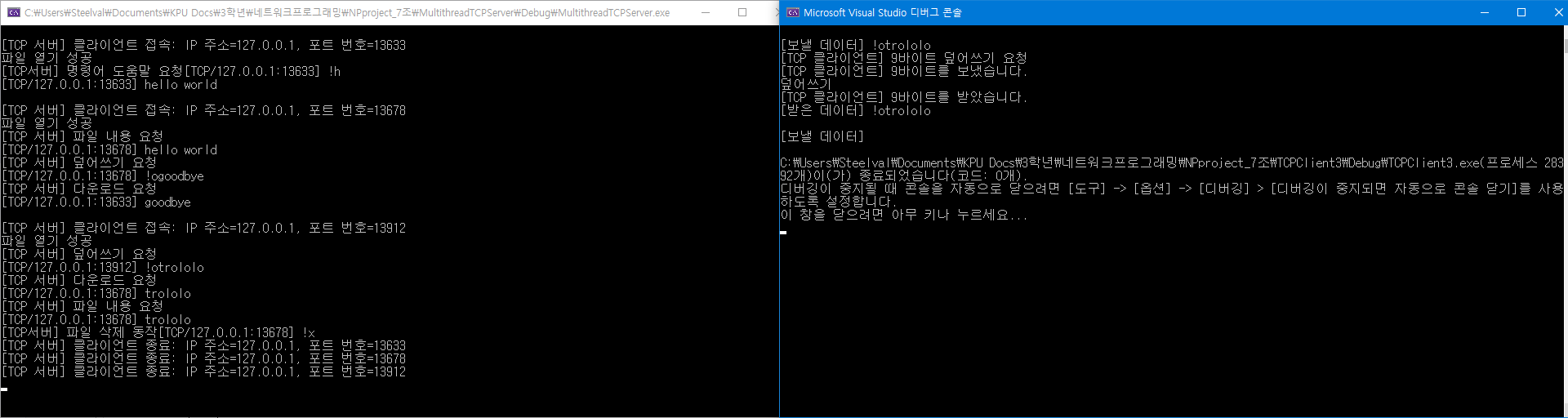
2번 클라이언트 측의 logcpy.txt에 해당 내용이 다운로드 된 모습이다.

6. 2번 클라이언트에서 다시 한번 서버 측 텍스트파일인 log.txt의 내용을 확인한 모습이다. 여전히 3번 클라이언트에서 변경한 내용 그대로이다.

7. 2번 클라이언트에서 자신의 파일의 내용을 삭제하였다.

2번 클라이언트의 logcpy.txt 내의 내용이 삭제된 모습이다.

8. 1번 클라이언트가 보낼 데이터에 아무 것도 입력하지 않고 데이터를 보내 접속을 종료한 모습이다.

9. 2번 클라이언트도 접속을 종료한 모습이다.

10. 마지막으로 3번 클라이언트도 접속을 종료한 모습이다.

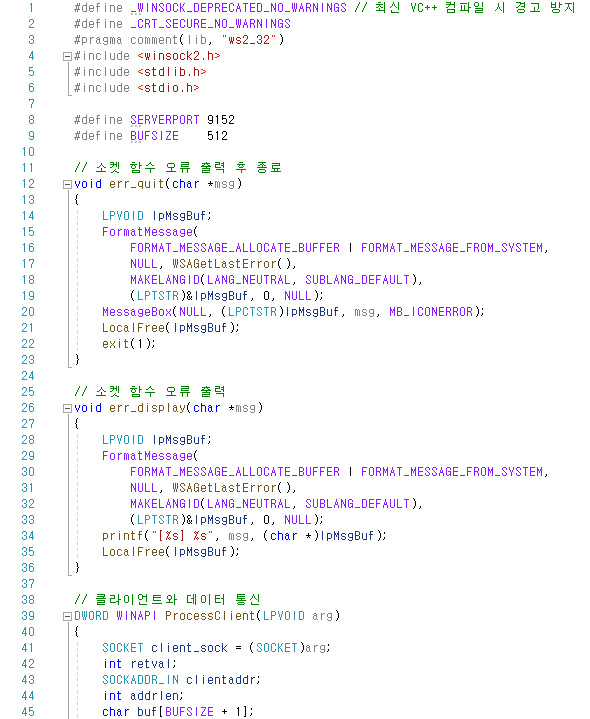
**4. 팀워크**

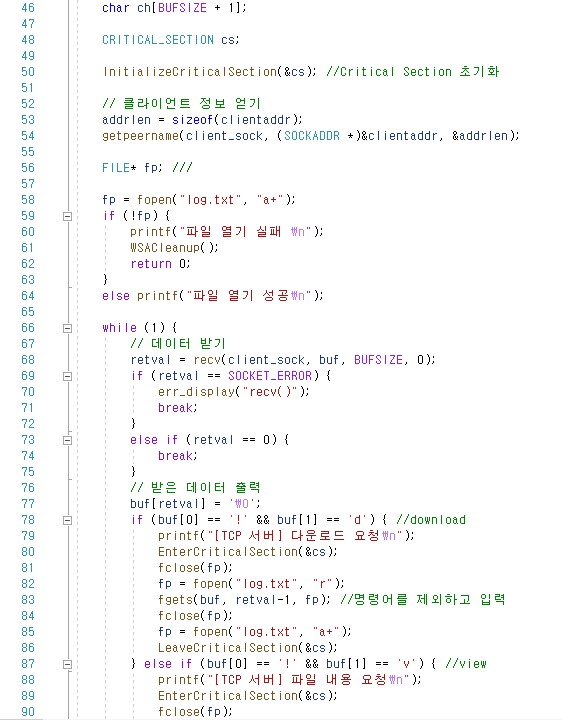
프로그램의 구상부터 제작까지 모든 작업을 혼자서 진행하였다.

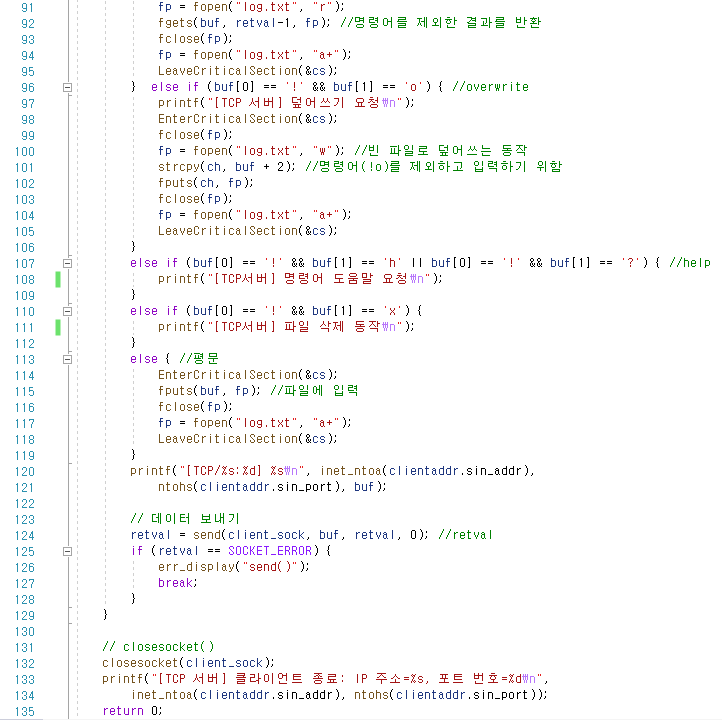
**5. 결론 및 총평**

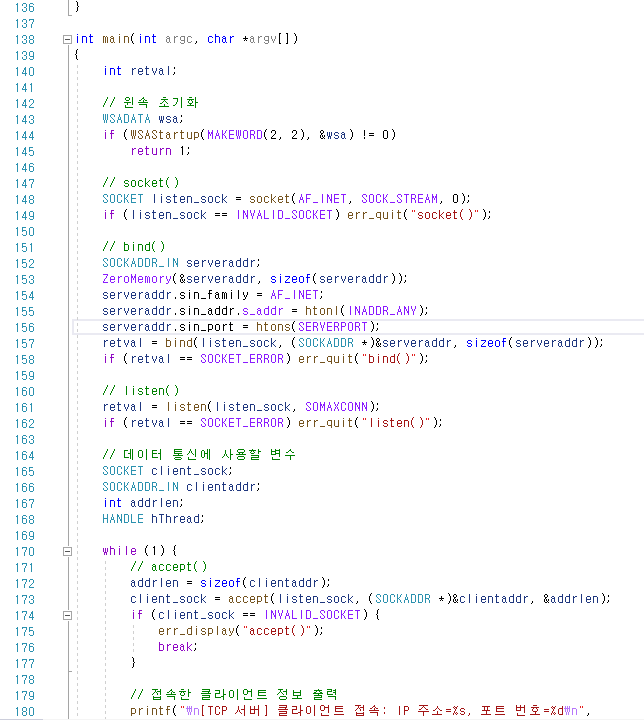
처음에 구상한 만큼은 아니지만, 핵심 기능과 더불어 어느 정도의 추가 기능으로 생각했던 부분까지 구현되었다. 해당 기능은 실험 결과 문제없이 동작한다. 파일 입출력 기능도 서버와 클라이언트 모두 동작함을 확인하였다. 기능이 얼마 되지 않는 줄 알았지만, 데모를 할 때 보여야 할 것이 생각보다 많다는 것을 알았을 때, 꼭 그렇지만은 않다는 사실을 깨달았다. 과제 시작 극 초반부에는 실패하면 어떡하나 하는 걱정도 했지만 끝나고 나니 처음에 생각했던 것보다 결과가 괜찮은 것 같아서 의미 있는 설계과제가 된 것 같은 느낌이 든다.

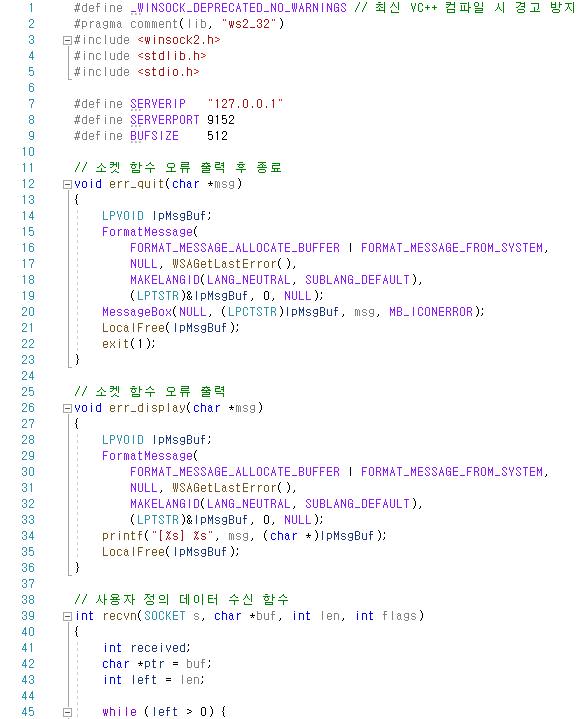
**6. 부록: 소스 코드**

- 서버(MultithreadTCPServer.cpp)







-클라이언트(TCPClient.cpp)

