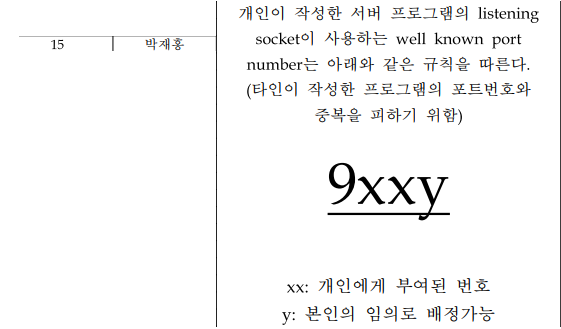
**소 속: 컴퓨터공학과**

**성 명: 박재홍**

**학 번: 2016150019**

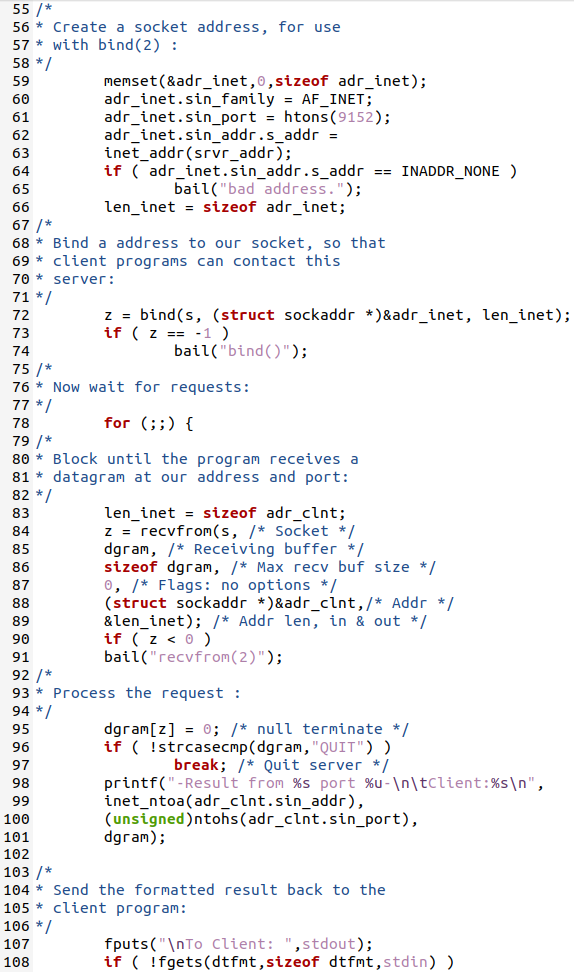
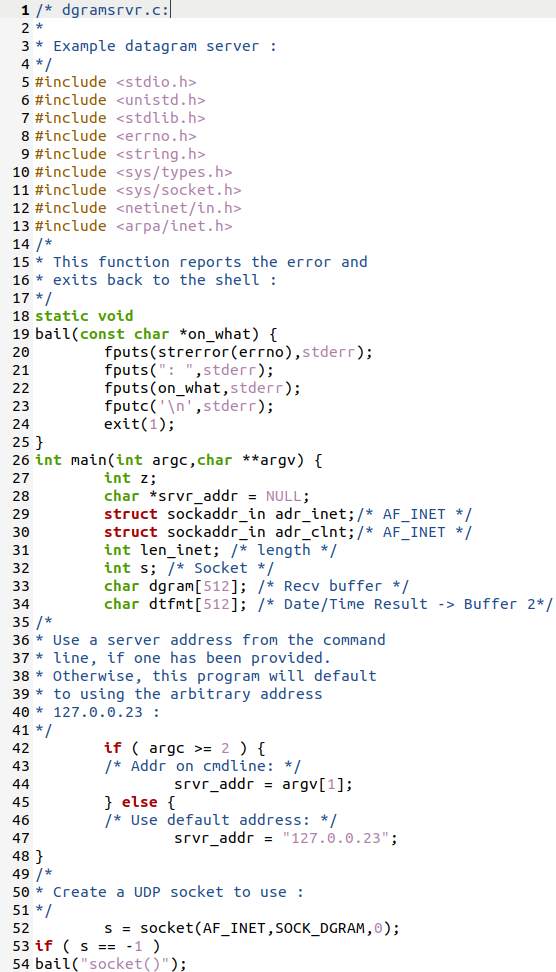
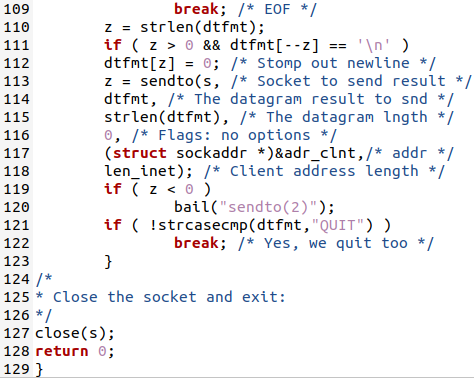
**제출일: 2020.05.06**

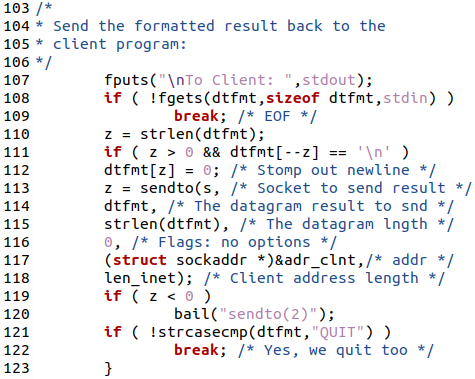
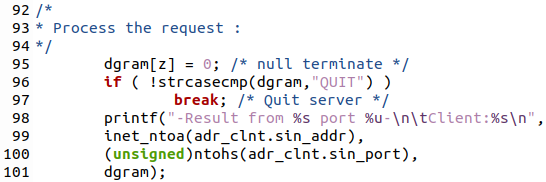
**네트워크프로그래밍 설계과제물 보고서(Two way chat)**

 본 프로그램을 만들면서 이용한 환경은 VMWare(버전 15)를 이용, Ubuntu 20.04 운영체제를 설치한 UNIX 기반의 환경이다. 해당 환경에서 프로그램을 만들게 된 이유는 이렇다. 프로그램의 대부분은 강의자료에 제시된 대부분을 참조(그대로 이용했다고 하는게 더 맞을 것이다)하였다. 그러나 Windows의 Visual Studio 혹은 Dev C++등의 환경에서는 강의자료에 제시된 코드가 동작하지 않았다. 혹시나 하여 인터넷의 도움을 받고자 하였지만, 애초에 include문에 있는 헤더 파일 자체가 없다고 하는 오류가 나는 경우가 많았다. 더 조사를 하니 오류가 발생한 헤더 파일들은 UNIX 기반의 시스템 환경에서는 여전히 이용하지만 Windows 환경의 Visual Studio의 경우는 Winsock(윈도우 소켓) 등으로 대체하여 해당 헤더 파일들을 굳이 가지고 있지 않았던 것이었다. 처음에는 Winsock이 앞의 헤더를 대체할 수 있다는 정보를 입수하여 그것을 이용한 방법을 이용해 보고자 하였으나 실상은 그렇지 않았고, 이용하는 함수, 코드가 전혀 달랐다. 따라서 UNIX 기반의 환경을 이용하기로 하였고, 결론은 이것이 옳은 선택이었다. 코드 자체는 강의자료를 대부분 참조한 것이 맞지만 강의자료에 나온 코드는 Echo server 및 client의 코드이고, 본 프로그램은 Server와 Client가 양방향으로 소통 및 프로그램 종료를 할 수 있도록 그것을 수정한 것이다. 주석은 프로그램의 어떤 부분이 어떤 역할을 하는지 구분이 잘 되도록 하여 굳이 지우지 않았다. 먼저 전체 코드를 캡쳐하여 첨부하지만 수정된 부분 위주의 설명을 할 것이다. 그리고 해당 환경에서 정상 동작함을 증빙하는 캡쳐 사진도 첨부한다.

Server 프로그램의 listening socket이 사용하는 well known port number는 첨부된 규칙에 따라 9152로 설정하였다. UDP 즉, Connectionless Protocol을 이용한 프로그램을 만들었다.

다음은 server 프로그램의 소스 코드 전문이다. 기본적으로 프로그램은 하나의 인자, IP 주소를 받을 수 있고, 입력된 IP주소에 9152포트를 이용하여 Listen 및 메시지 송수신이 가능하도록 하도록 하고, 기존 프로그램과 다르게 server 측에서도 프로그램 종료(server 및 client)가 가능하도록 하였다. 해당 기능 외에 불필요하다고 판단된 코드 및 헤더 파일 include문은 삭제하였다.

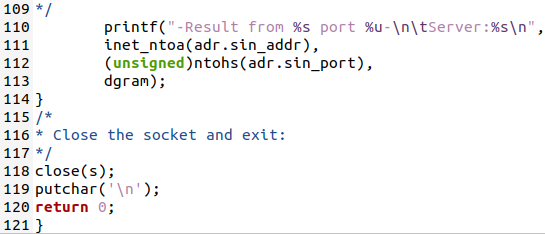
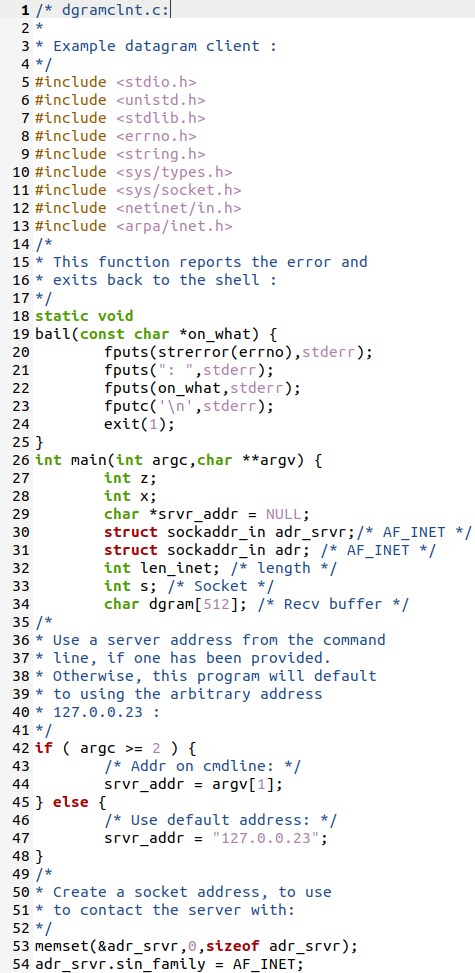
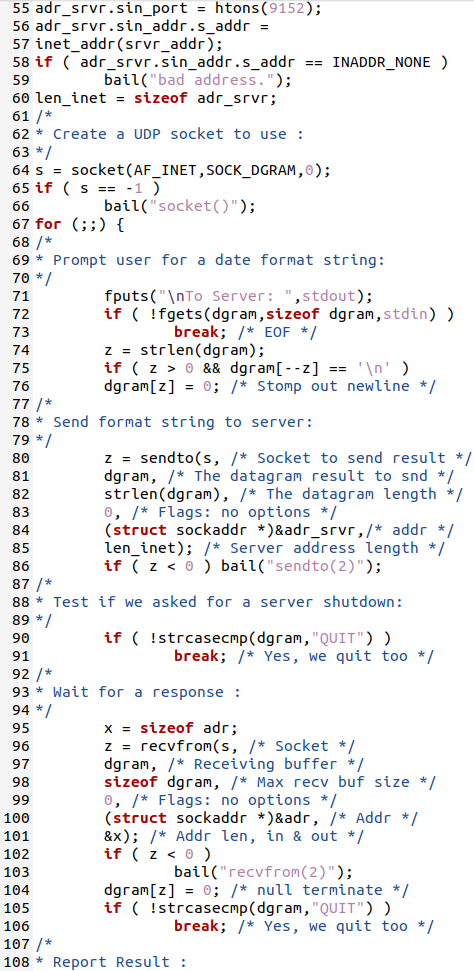


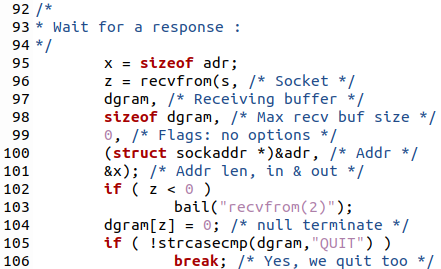
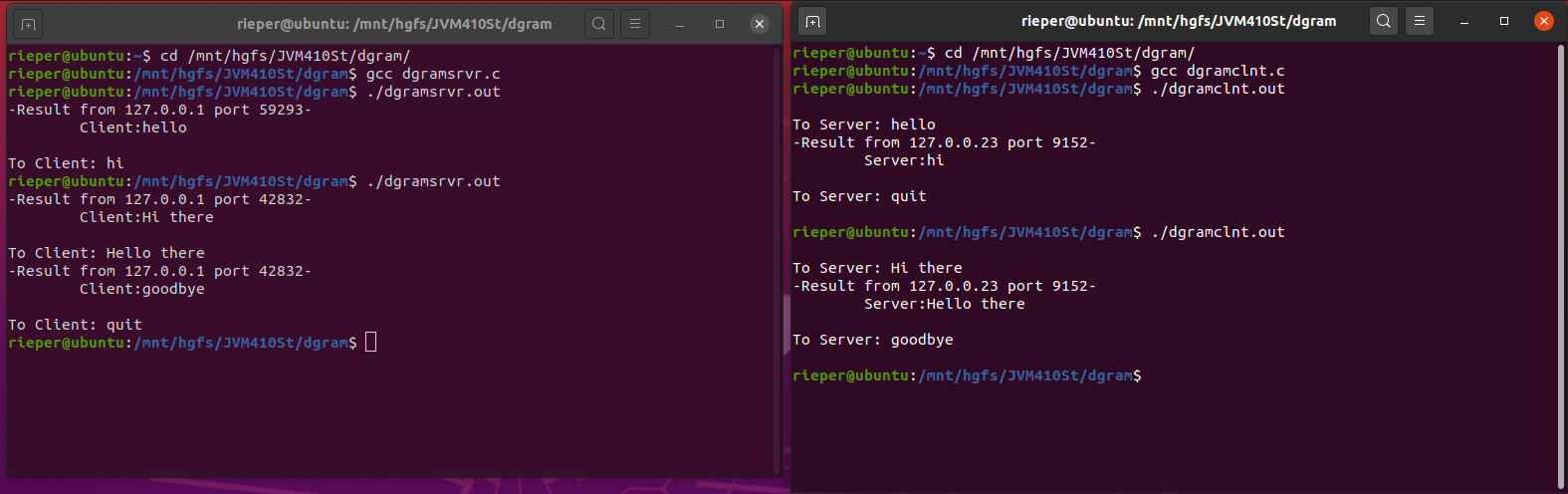
 위 코드는 기존 강의 자료와 구분 지을 수 있고 프로그램의 핵심 기능 중 하나인 양방향 소통 중 한 부분으로, 사용자에게 보일 인터페이스(혹은 메시지)를 제외하고 변경된 가장 중요한 부분이다. 먼저 기존의 dtfmt 변수를 메시지 전송에 이용하도록, 버퍼로 수정하였다. 서버는 기본적으로 client 측에서 메시지를 보낼 때까지 대기하게 되는데(해당 코드 이전 부분에 구현), 그렇게 받은 메시지를 먼저 표시하고, 답문을 보낼 수 있도록 하였다. 이 프로그램이 server 측임을 더 잘 구분되도록 전송 시 'To Client'라는 표현을 사용하였다.

프로그램을 종료하는 경우는 두 가지가 있다. 첫 번째는 Client 측에서 대소문자 관계없이 'quit'을 입력하는 것이고, 두 번째는 Server 측에서 그렇게 하는 것이다. 먼저 메시지 수신 버퍼로 이용된 dgram 변수는 대소문자에 관계없이 'quit'이 입력되면 반복문을 탈출(즉, 프로그램을 종료)하도록 한다. 다시 말해, client 측에서 'quit'을 입력하면 그 메시지를 server 측으로 전송하게 되고, server 측은 그에 따라 프로그램을 종료하게 되며, client 측에서도 (메시지를 전송한 다음)프로그램을 종료하게 된다. 이는 곧 client 측에서 server 측에 'quit' 메시지를 보내는 경우에 해당한다.

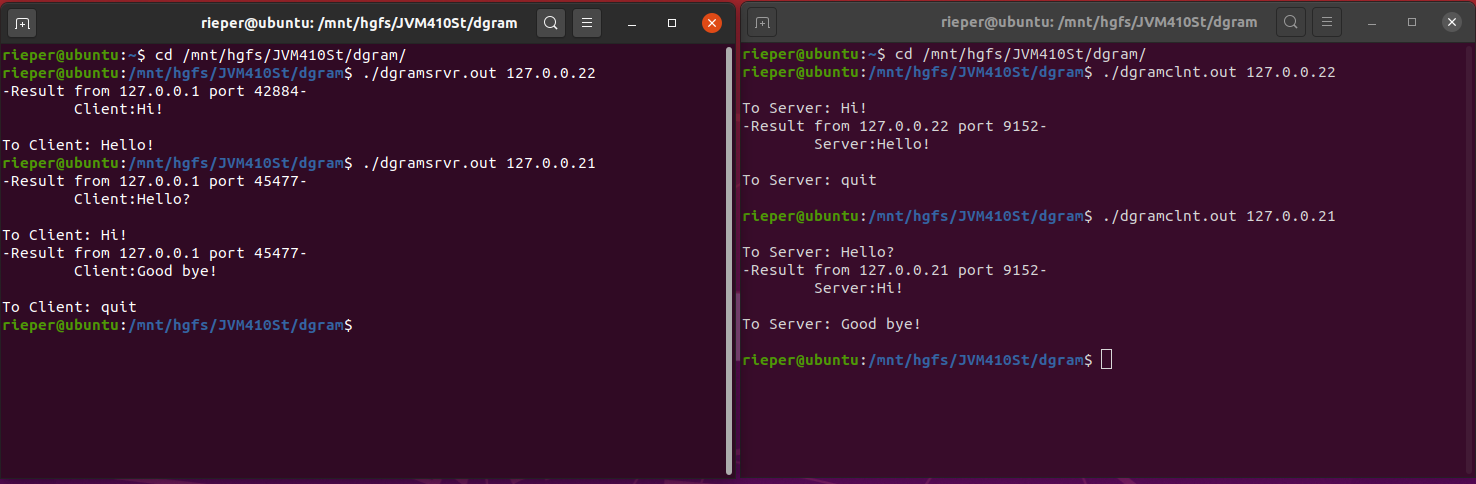
반대로, server 측에서 client 측에 'quit' 메시지를 보내는 경우, 메시지의 송신 버퍼로 이용된 dtfmt 변수도 같은 역할을 한다. 그것도 후술하겠지만 client 측의 버퍼에서도 같은 역할을 하는데, 'quit'이 입력되면 먼저 해당 정보를 client 측에 전송하여 상술한 것처럼 해당 프로그램을 종료하게 하고, server 프로그램도 마찬가지로 종료하게 된다. 결론적으로 어느 쪽에서 quit을 입력하더라도 두 프로그램 모두 종료된다. 기본적으로 server는 client로부터 메시지를 받는 것으로 프로그램이 시작하며, 그 전까지 대기한다. 따라서 메시지를 받을 때와 보낼 때 종료하는 부분을 모두 구현해야 한다.

다음은 클라이언트 측의 소스 코드 전문이다. 기본적으로 서버 프로그램과 같이 메시지 송수신 및 프로그램 종료가 가능하다. 연결할 IP주소를 프로그램의 인자로 받는 것도 서버와 같다. 위의 서버도 마찬가지지만 인자를 입력하지 않으면 디폴트로 127.0.0.23 주소에 연결하도록 한다.



Client 측에서 기존의 프로그램과 비교하여 달라진 가장 중요한 부분은 바로 이 부분일 것이다. 기존과 대비 달라진 부분이 많지는 않다. 기본적으로 메시지를 보내고 받는다는 개념은 동일했기 때문이다. 따라서 일부 변수만 수정하여 양방향 메시지 송수신을 구현할 수는 있었다. 그러나 기존에는 물론 client 측에서만 메시지를 보냈기 때문에 종료도 client 측에서만 가능했으나, 이제는 양방향 chat이기에 둘 중 어디에서나 프로그램 및 연결 종료를 할 수 있어야 한다고 생각하여 그것을 구현한 것이다. 기본 원리는 위에서 설명한 것과 같다. 만약 server 측에서 'quit'이라는 메시지를 보낸다면, 그것을 버퍼에 저장하고 그것을 strcasecmp 함수를 이용하여 대소문자에 관계없이 'QUIT'이라면 반복문을 탈출 및 프로그램을 종료하게 된다. 메시지를 보내는 부분에서의 프로그램 종료는, 기존 프로그램에 구현되어 있어 수정할 필요가 없었다.

프로그램 실행은 UNIX 계열 환경에서 Terminal을 통하여 './' 명령어를 이용하여 실행하면 된다. 위 캡쳐된 이미지는 server 및 client(왼쪽부터 순서대로 server, client) 모두 각 프로그램을 실행하여 메시지를 보내고, 첫 번째 실행에서는 client측에서 'quit'을 입력하여 양측 모두의 프로그램을 종료시킨 상황이고, 다음 실행에서는 server 측에서 'quit'을 입력하여 양측 모두의 프로그램을 종료시킨 상황이다. 모두 정상 동작함을 알 수 있다. 인자를 입력하지 않아 server 측의 IP가 127.0.0.23임을 확인할 수 있다.

 다음은 프로그램 실행 시 인자로 IP주소를 입력했을 때의 결과이다. Client 측에서의 메시지를 통해 입력한 인자대로 server 측의 IP주소가 설정되었음을 알 수 있고, 양측 중 어디에서 'quit'을 입력하더라도 정상적으로 양측 모두 프로그램이 종료되는 모습을 확인할 수 있다.