**Project #3: P2P File Transfer**

9팀

2016146007 김성결

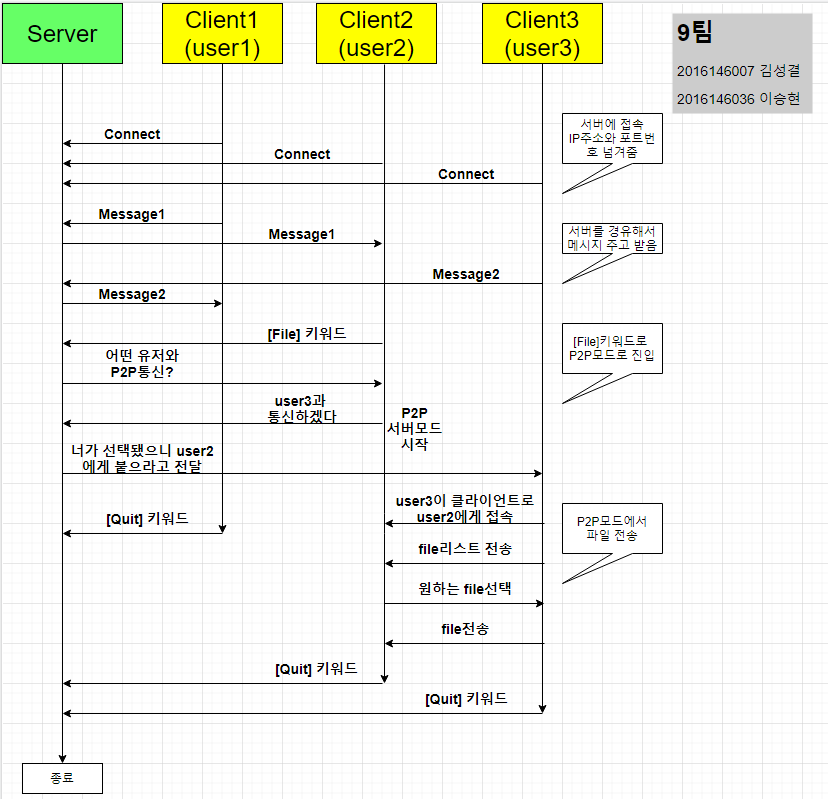
2016146036 이승현

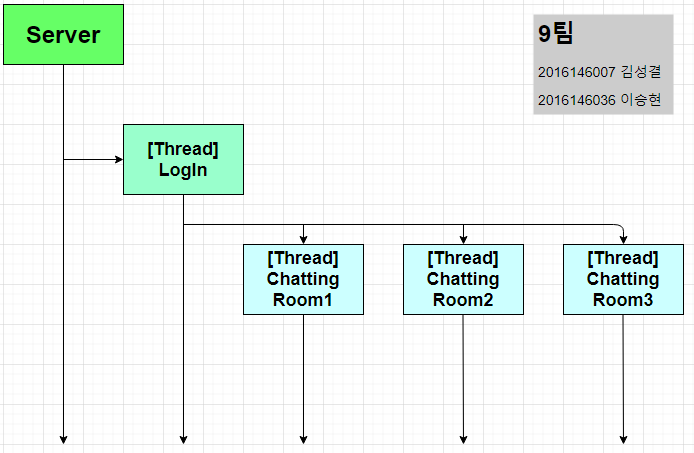
■기능 구현

1. 클라이언트 최대 3명이 동시에 채팅 가능
2. #quit 키워드로 클라이언트 한 명씩 로그아웃(접속종료) 가능
3. #exit 키워드로 클라이언트 전부 동시에 로그아웃(접속종료)가능
4. [File] 키워드로 다른 클라이언트와 P2P모드로 진입. 상대 클라이언트의 파일 목록 받아와서 전송받고 싶은 파일 선택하면 수신 가능

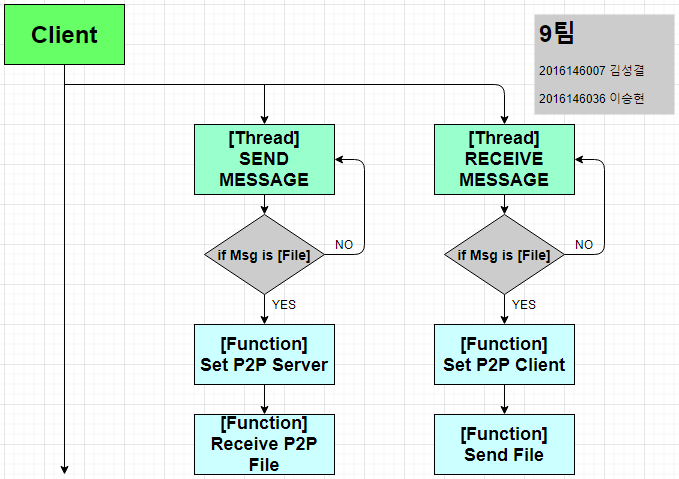
※이때 기능구현에선 다양한 병렬프로세스를 실행시키기 위해서 fork대신 **쓰레드를 사용**

■프로그램 구조

**1.** Flow Chart

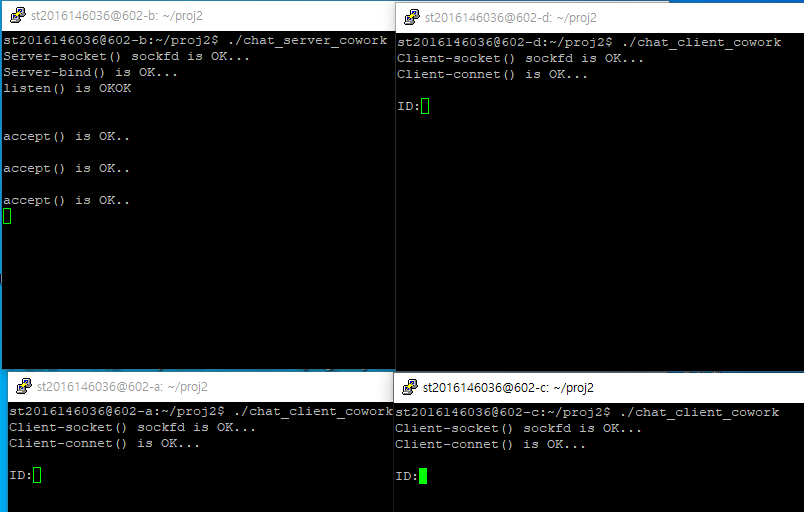
**2.** Server Structure

쓰레드를 사용해 동시에 여러 클라이언트를 받을 수 있게 코드를 작성**.**

**3.** Client Structure

[File]이란 특수명령어가 아니면 계속 해서 메시지를 보내거나 받고 [File]이란 명령어가 들어오면 P2P함수를 작동시킴 (함수를 쓴 이유는 p2p작업할 때 다른 기능을 멈추게 하기 위함)

■동작 사진

**1. 서버에 다수 클라이언트 접속**

클라이언트(101))

서버(103)

클라이언트(101)

클라이언트(102)

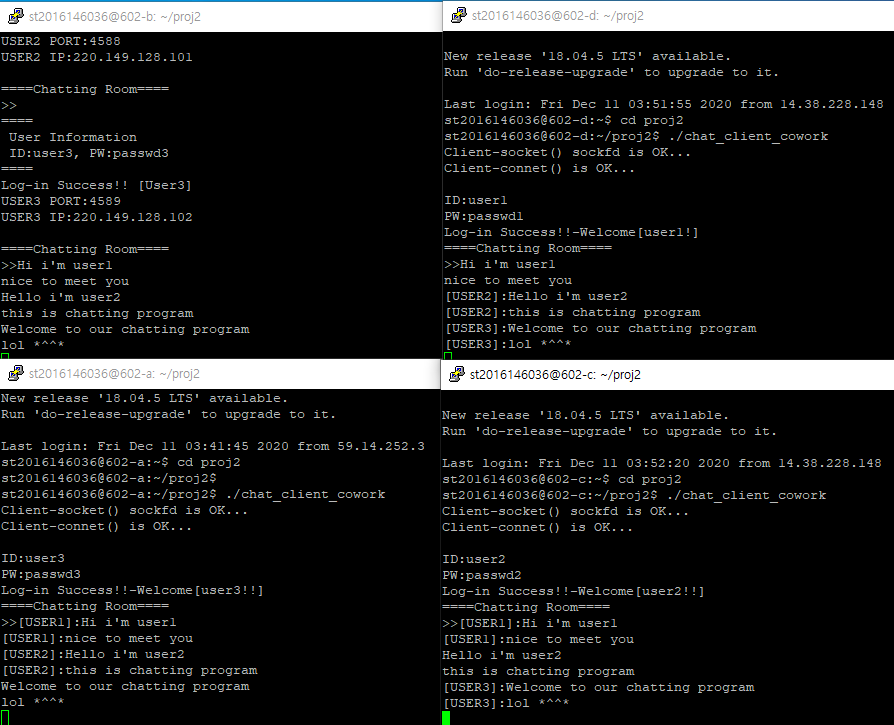


클라이언트(102)

클라이언트(100)

▲해당 부분의 서버 측 코드

다수의 클라이언트를 접속시키기 위해 while(1)안에다 accept를 받아주었으며 accept가 들어오면 쓰레드로 로그인 프로세스가 진행되도록 구현하였다. 이때 new\_fd는 쓰레드에서도 사용할 수 있도록 전역변수로 설정함.

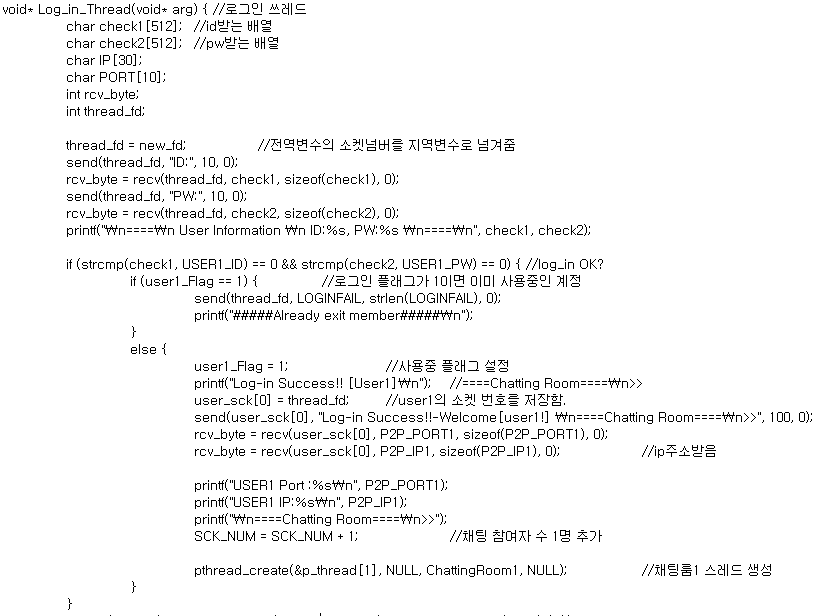
**2. 다수 클라이언트 로그인 후 채팅방 진입**

user3

user2

user1

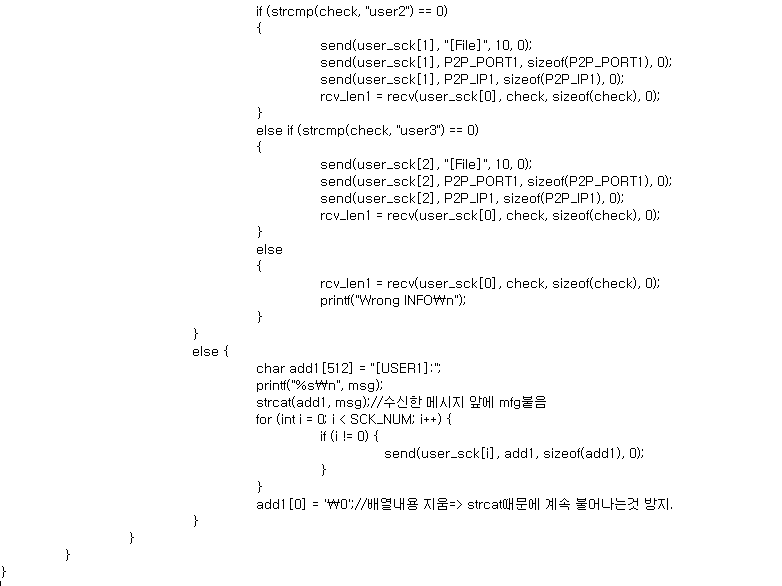
Server

▲서버에서 로그인 시도한 클라이언트의 정보(IP주소, 포트번호)를 확인할 수 있다. 로그인에 성공하면 채팅방 쓰레드로 진입한 후 단체 채팅 시작. (로그인 시 서버가 받은 **IP주소와 포트번호는 저장된다**) 사진에서 볼 수 있듯 채팅을 치면 다른 클라이언트로 전송이 되는 것을 볼 수 있다.

▲서버 측 로그인 스레드. strcmp로 확인한 로그인 정보가 일치하면 채팅방 스레드 생성.

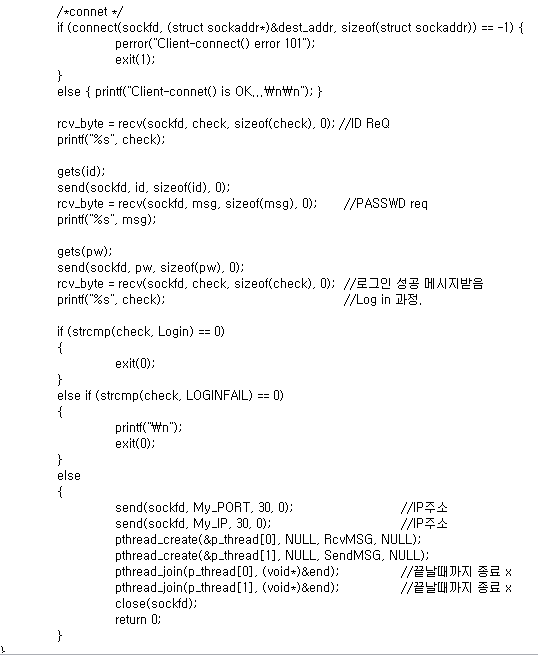
로그인 시 전달받은 IP와 포트 번호는 P2P\_PORT, P2P\_IP 전역변수에 저장되고 추후에 사용된다.

이때 전역변수로 받은 new\_fd를 쓰레드의 지역변수로 받아주는데 이는 다중 접속을 할 때 new\_fd가 갱신되기 때문이다. 즉, 쓰레드에서 사용하고 있던 new\_fd값이 중간에 변하면 send, receive 작업중 소켓 번호가 바뀌기 때문에 지역변수로 받아주어 관리하도록 구현하였다.



▲서버 측 채팅방 스레드 (전체)

while(1)에서 send와 receive가 반복된다. **빨간색 타원**의 코드를 통해서 한 클라이언트가 보낸 메시지를 본인을 제외한 다른 클라이언트들에게 모두 전달해준다. SCK\_NUM은 현재 접속 중인 클라이언트의 수로 유저가 한 명 로그인 할 때 마다 1씩 증가하는 값이다. user\_sck[i]은 각 인덱스마다 사용 중인 소켓 번호가 저장돼 있다. 즉 user\_sck[0]에는 user1의 소켓번호 값이, user\_sck[1]에는 user2의 소켓번호 값이 들어가 있다.



▲클라이언트 측 코드

클라이언트측은 Connect가 되면 id와 pw를 전송한 뒤, 로그인 성공 메시지를 받았다는 것을 확인했으면 자신의 IP주소와 PORT번호를 서버 측에 전달해준다. 그 후 메시지를 받는 쓰레드인 RcvMSG와 메시지를 보내는 쓰레드인 SendMSG를 실행시킨다. 쓰레드가 끝날 때까지 프로그램이 종료되지 않도록 pthread\_join으로 쓰레드의 종료를 기다려준다.

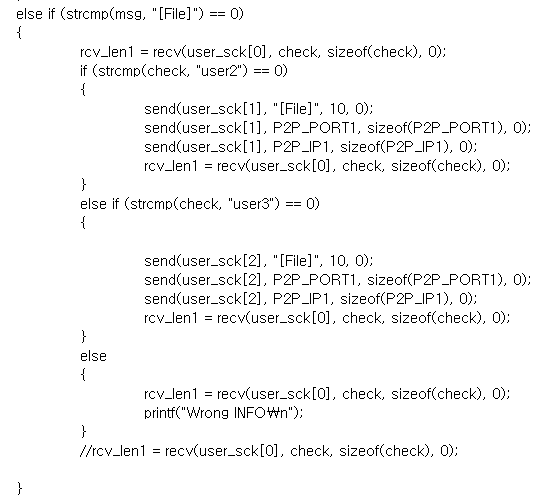
◀SendMSG 쓰레드

◀RcvMSG 쓰레드

이렇게 두개의 쓰레드로 나눈 이유는 실시간으로 채팅이 오가기 때문에 메시지를 받으면서 채팅을 칠 수 있는 상태여야 하기 때문이다.

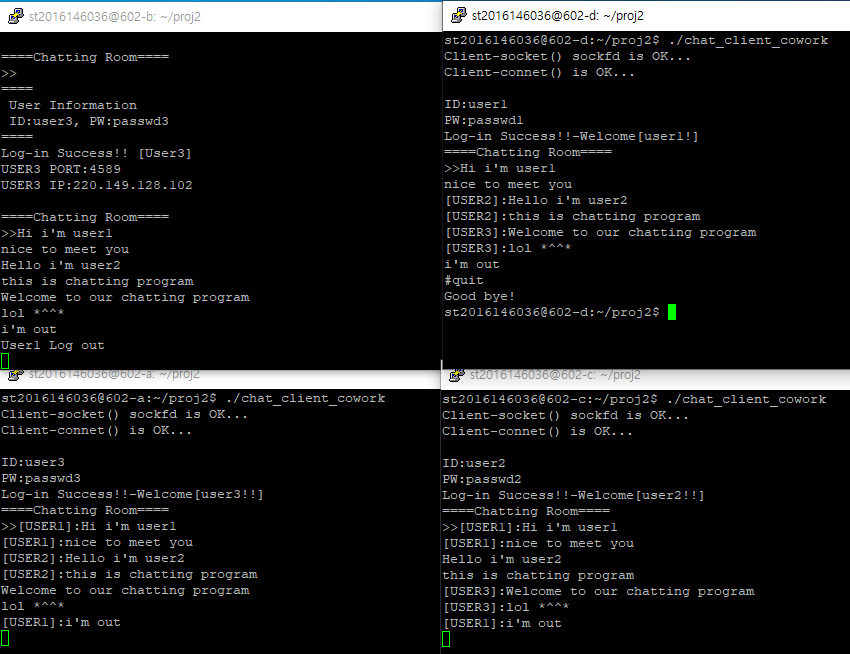
SendMSG코드에서 filemode라는 것은 해당 클라이언트가 [File]이란 명령어를 보냈을 때 P2P실행모드로 들어가고 채팅 메시지를 send하는 작업을 실행하지 않기 위해 설정해 주었다.

strcmp로 자신이 서버로 보낸 메시지가 [File]이라는 명령어라면 즉시 어느 유저로 보낼지 gets함수로 입력 받고 해당 유저이름을 서버로 보내준다. [File]이 아니라면 메시지를 그대로 전송해준다.



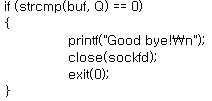
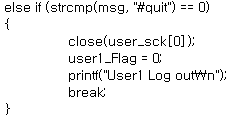
▲서버측 user1 채팅방 쓰레드 (일부분)

[File]이란 명령어를 받은 서버측 채팅방에서는 user의 이름을 확인하고 해당 유저의 소켓으로 [File]이란 명령어와 [File]을 보낸 유저의 포트번호와 IP번호를 전송해준다. (여기서는 user1의 포트번호와 IP주소를 보냄) **이 부분은 보고서 12페이지 “[File] 키워드를 통한 P2P모드 파일 전송”에서 구체적으로 서술하였다.**

**3. #quit 키워드를 통한 개별 로그아웃**

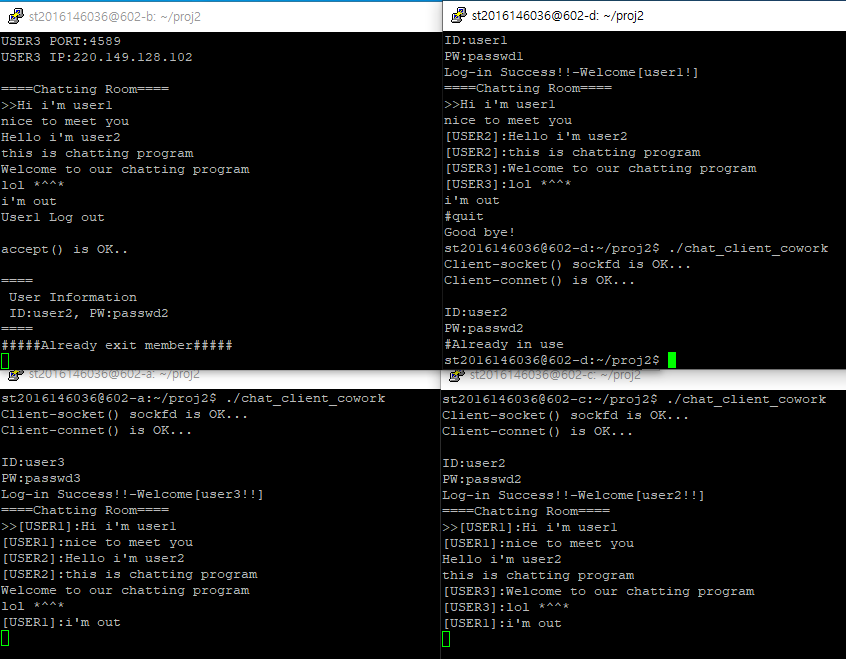
Server

user1

▲**user1이 로그아웃** 했다는 정보가 서버에도 출력됨. user2와 user3은 접속 중. 

▲서버 ▲클라이언트(Q는 “#quit”를 담고있다)

서버 측의 채팅방 쓰레드를 보면 strcmp에서 특수한 문자열들을 비교하는 문항이 있는데 해당 문자열들을 클라이언트 측에서 보내면 서버가 소켓을 닫고 접속이 아니라는 플래그를 설정하도록 코드를 작성하였다. (user1\_Flag는 전역변수로 해당 값이 0이면 접속이 아니라는 의미)

**4. 중복 로그인 방지**

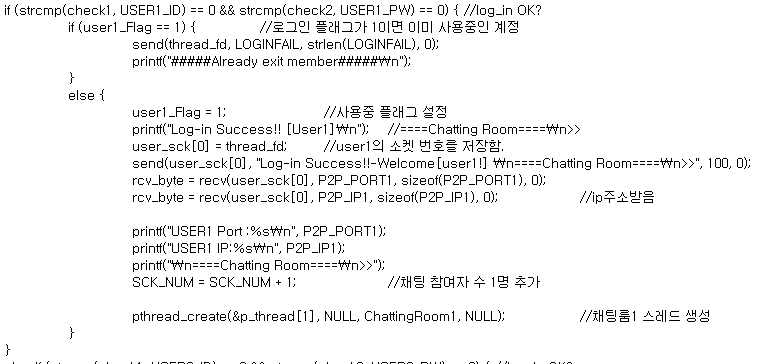
Server

user2

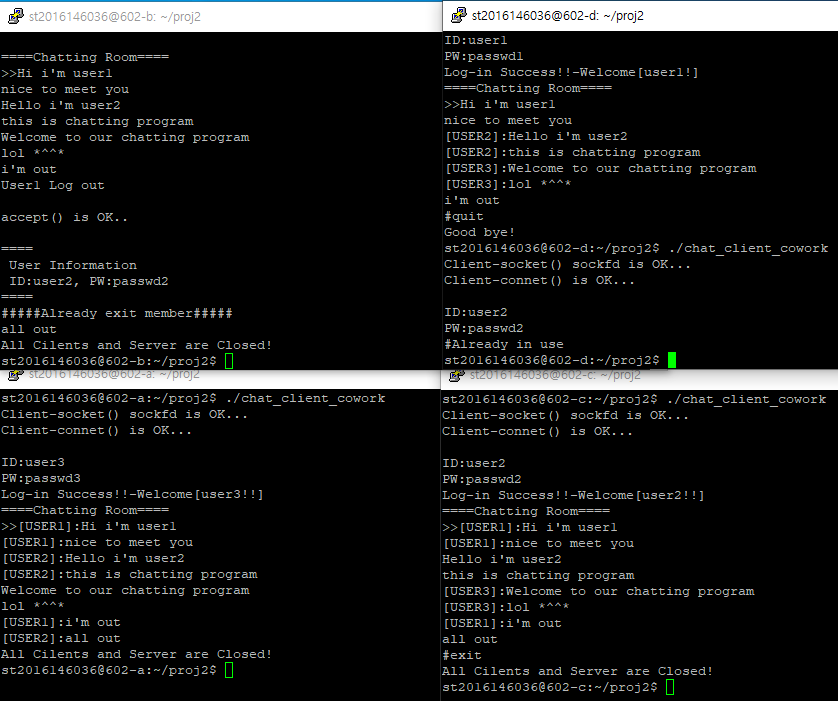
중복!

user2는

이미 로그인 중

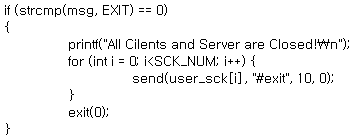
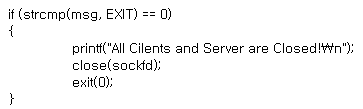
▲user1이 개별 로그아웃한 100클라이언트에서 user2로 로그인을 시도하면 서버와 클라이언트 측에 **이미 사용 중인 계정**이라고 경고가 뜨며 로그인에 실패 (#Already in use라는 경고문 출력)

▲서버 측 로그인 스레드의 일부이다. 로그인 시 플래그 값을 확인하여 0일 경우에만 로그인이 가능하고. 로그인 후 1로 설정.

**5. #exit키워드를 통한 모든 클라이언트와 서버 종료**

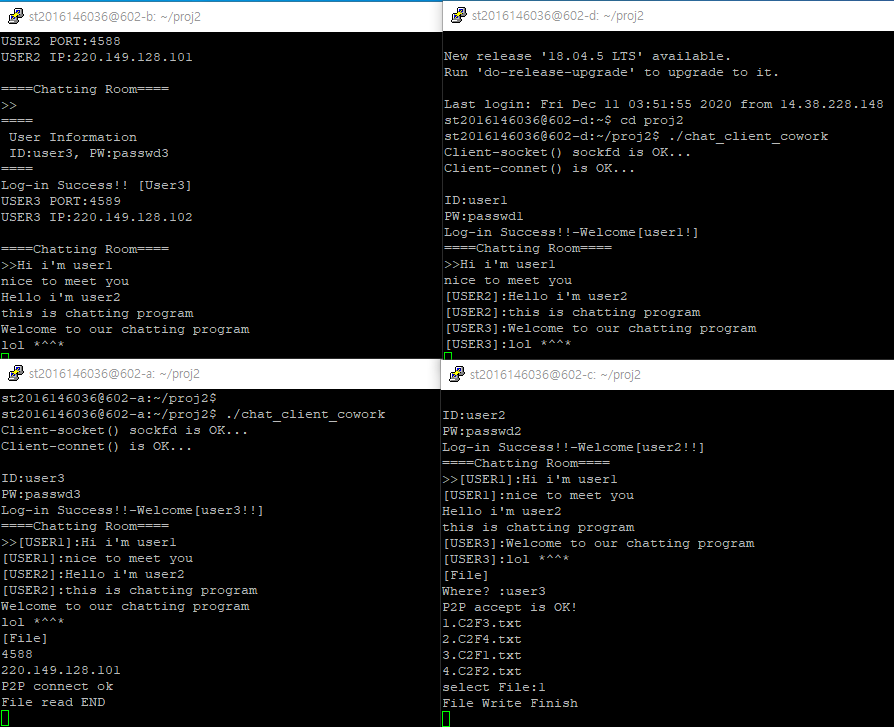
user2

▲user2에서 #exit를 통해 **모든 클라이언트와 서버를 종료시키는 모습.**

◀서버◀클라이언트

서버에서 #exit를 받으면 모든 클라이언트에게 키워드를 전달한 후 종료하고 클라이언트들도 메시지를 받은 후 모두 종료되는 모습이다.

**<Project 3 내용>**

**6. [File] 키워드를 통한 P2P모드 파일 전송**

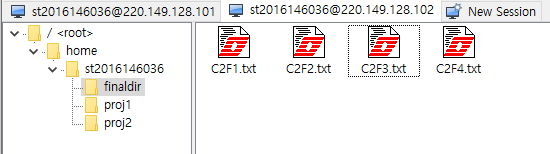
Server

user1

user2

user3

▲user2가 [File] 키워드를 입력하여 user3를 선택함. [File]키워드를 전송한 user2는 P2P모드에서 서버로 작동하고 user3이 클라이언트로 작동하게 코드를 작성함.

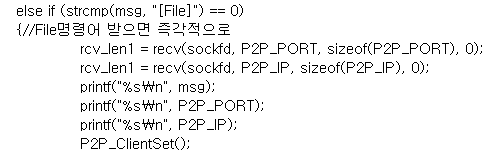
위의 사진에서는 user3이 자신의 파일 목록을 전송해주면 user2가 **하나를** **선택**하여 전송 받는 모습이다.

▲102(user3)의 파일 리스트

여기서 핵심은 모든 클라이언트가 이미 생성된 **디렉토리(finaldir)에 전송용 txt파일을 미리 가지고 있다는 점이다. (경로는 /home/st2016146036/finaldir) 즉, 다른 디렉토리의 파일을 옮기려면 코드에서 디렉토리 관련 경로를 바꿔줘야 한다.**

◀SendMSG 쓰레드

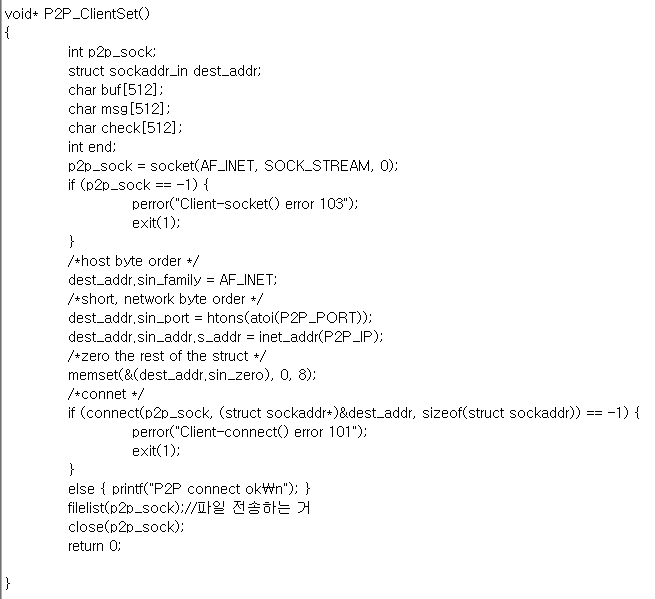
Send MSG쓰레드에서 File명령어를 입력후 제대로된 유저번호를 입력했다면 P2P\_ServerSet()이라는 함수 실행.

◀RcvMSG쓰레드중 일부

RcvMSG쓰레드에서 만약 클라이언트가 [File]이란 명령어를 받았다면 즉각적으로 [File]이란 메시지를 보낸 유저의 포트 번호와 IP주소를 받아와 전역변수에 저장한뒤, P2P\_ClientSet()이라는 함수 실행.

P2P와 관련된 코드들을 함수로 만든 이유는 앞에서도 설명했듯 P2P작업중에는 다른 작업의 영향을 받지 않도록 하기 위함이다.

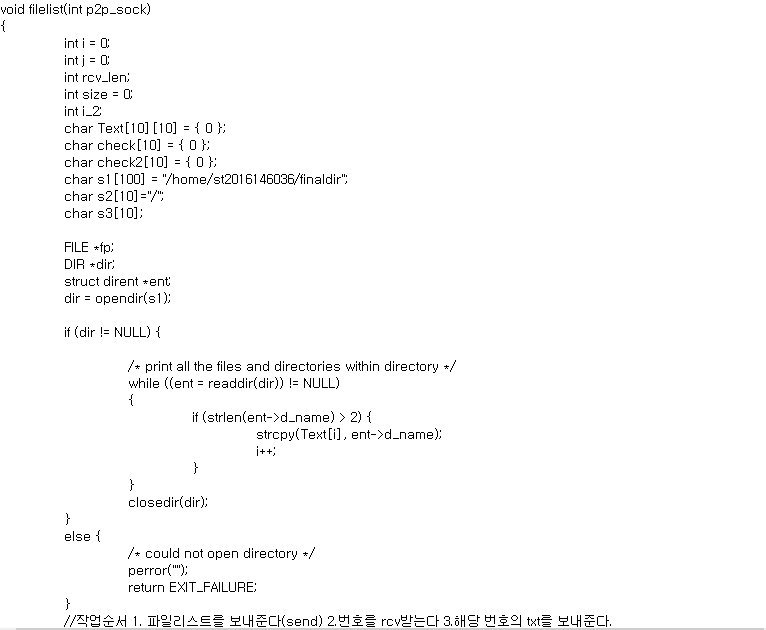
◀P2P\_ServerSet() 함수

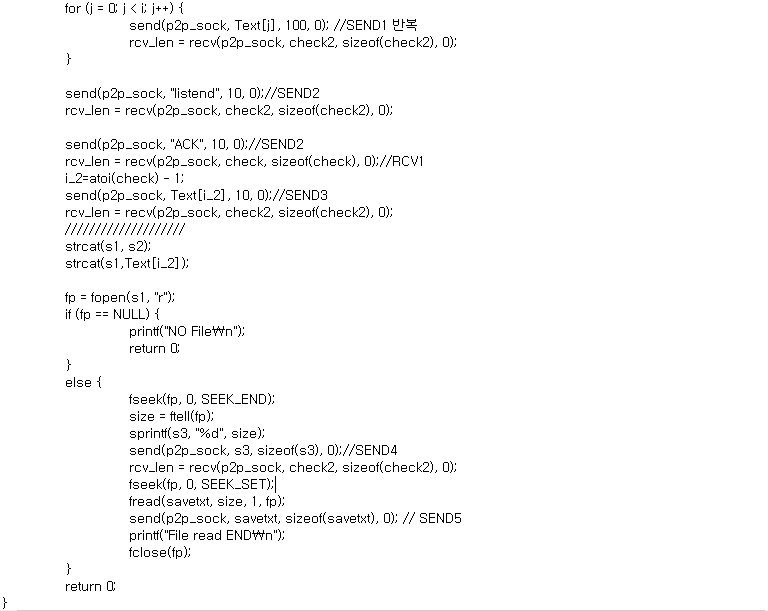
◀P2P\_ClientSet() 함수

위 코드는 **P2P\_ ServerSet()과 P2P\_ ClientSet()함수로** 둘다 기존의 서버만들기, 클라이언트 만들기 코드에서 크게 변한 점은 없다. 한가지 다른점은 포트 번호와 ip주소를 rcv즉 char형 배열로 받는 다는 점인데 IP주소는 상관이없지만 포트번호는 int형을 사용해야되므로 atoi()명령어를 써서 P2P\_PORT값을 정수형으로 바꿔주었다.

그리고 서버와 클라이언트가 서로 연결됐다면 각각 파일리스트를 받는 함수는 Rcv\_P2P함수와(서버측) 파일리스트를 읽고 해당 파일을 복사해주는 filelist()함수를 실행시켜 주었다.

**\*filelist함수**

◀filelist()함수 (1)

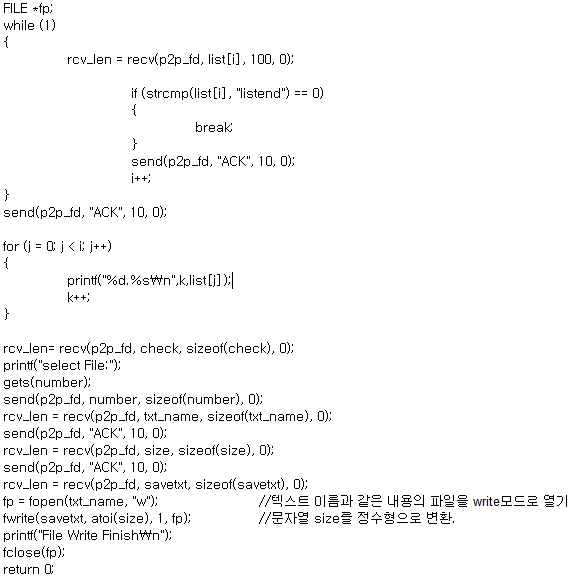
◀filelist()함수 (2)

앞선 코드는 filelist함수의 전체적인 코드이며 디렉토리 구조체과 파일 구조체를 사용했다.

코드는 우선 opendir로 지정된 디렉토리를 찾아간 다음 readdir명령어를 사용해 NULL이 아닐 때까지 해당 디렉토리에 있는 파일들을 읽어온 뒤, 해당 파일 이름을 Text[][] 2차원 문자열 배열에 저장했다. 그 후 해당 Text의 배열을 P2P\_Server에 전송한다. 전송이 완료되었으면 listen이라는 신호를 서버에 보내주고 몇 번째 파일을 받을지를 문자열로 받은 뒤 해당문자열을 atoi를 사용해 i\_2라는 int형으로 바꿔 Text[i\_2]번째 있는 파일 이름을 보낸다.

해당 파일의 내용을 복사해주는 코드가 2번 코드인데 파일 포인터를 사용해 해당 파일을 읽기 모드로 연 뒤 fseek로 파일의 포인터를 파일의 끝으로 보내준다. 이 상태에서 ftell명령어를 쓰면 파일의 size를 알 수 있는데, 파일의 size를 서버로 전송시켜준 뒤 fseek로 파일 포인터를 다시 처음으로 보내준다. 그리고 fread함수를 써서 size만큼의 문자열들을 문자열 배열 savetxt에 저장해주고 savetxt를 전송하면 파일의 복사가 완료된다.

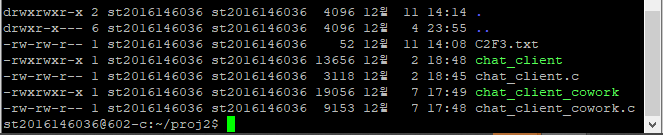
**\*Rcv\_P2P함수**

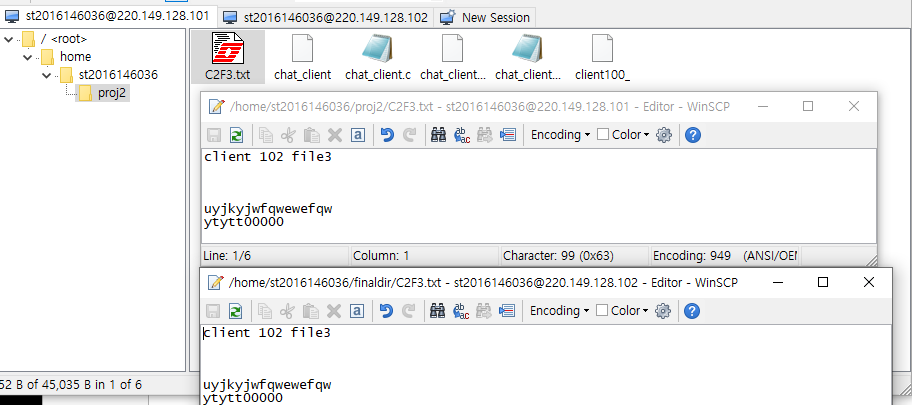


▲Rcv\_P2P함수에서 파일 리스트를 받아오는 로직

Rcv\_p2p함수에서는 파일 포인터를 사용하였으며 while(1)을 써서 p2p\_client에서 보내주는 파일의 이름을 list라는 배열에 저장했다. 그리고 listen이라는 파일을 다 보냈다는 문자열이 오면 while문을 나가게 설정했다.

파일 이름을 받는 작업이 끝났으면 for문으로 파일의 이름들을 print시켜준 뒤 gets함수를 써서 어떤 파일을 선택할지 고르고 해당 번호를 client에게로 전송한다. 그리고 client에게서 전송되는 파일의 사이즈와 txt의 문자열을 receive하고 fopen으로 파일 이름을 쓰기모드로 연 뒤 fwirte를 사용해 client에서 온 size와 텍스트 내용을 적어준다.

**\*파일전송결과\***

▲**위쪽** txt 파일이 101(user2)에서 **전송받은 파일**이고, **아래쪽** txt 파일이 102(user3)의 **원본 파일**이다. 내용까지 잘 전송된 것을 확인할 수 있다.