임베디드 운영체제

Final Project #3

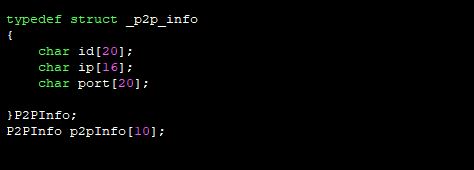
전자공학부 임베디드시스템 전공

2018148035 한혜지

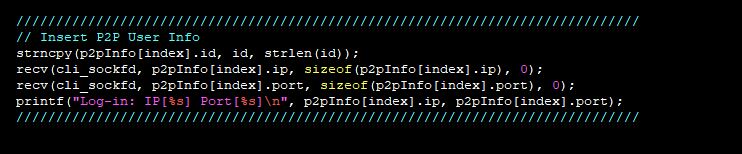
2018146036 황서영

1단계) 서버 - P2P 파일 전송

#Project3에서 #Project2에서와 같이 클라이언트들간의 채팅 메시지를 주고받는 기능 외에 추가로 파일을 P2P로 파일을 주고받는 Server와 Client 기능을 구현하였습니다. Server와 Client의 통신은 TCP를 사용하였습니다. 채팅 대화 중 특정 사용자에게 파일 리스트를 요청하면, 요청한 사용자는 Server를 담당하고, 요청을 받은 사용자는 Client를 담당하여 서버에 접속 후 파일 리스트와 P2P 서버로부터 요청 받은 파일을 Server에 전송합니다. 전송이 완료되면 두 Server와 Client는 접속을 종료합니다.

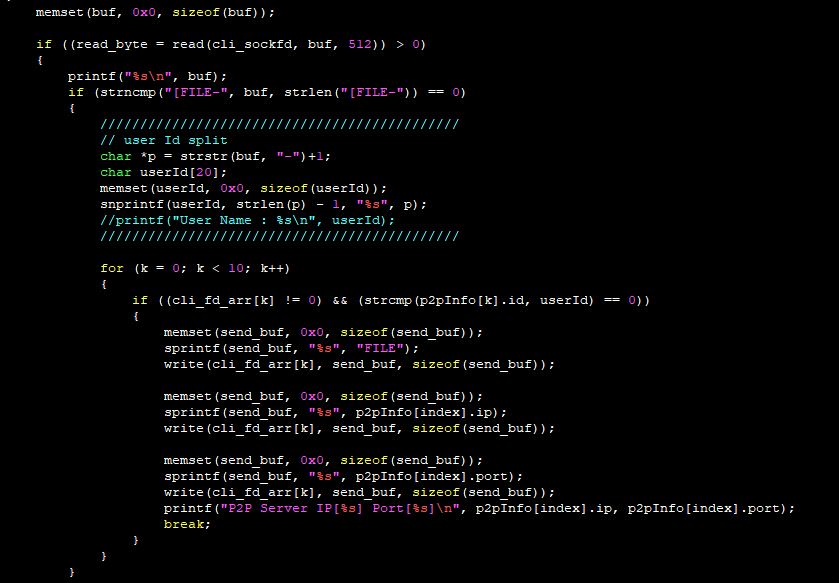


#Project3에서는 채팅 사용자간의 P2P 통신을 하기 위해서 P2P 접속 정보가 추가로 필요합니다. 서버는 P2P 접속 정보를 관리하기 위해 struct를 이용하여 P2P 접속 정보를 관리하였습니다. struct의 이름은 \_p2p\_info이며 typedef를 사용하여 P2PInfo의 별칭을 만들었습니다. P2PInfo는 전역으로 선언하였고, 최대 10명의 사용자 정보를 관리할 수 있도록 배열로 관리하였습니다. P2PInfo에서 관리하는 정보는 ID, IP, Port 입니다. ID는 Login시에 수신된 ID로 채팅 대화 중인 사용자가 특정 사용자를 지정하여 P2P 통신을 할 때 특정 사용자를 찾기 위한 목적입니다. IP와 Port는 P2P Server의 정보이며, 이 정보를 이용하여 접속을 시도하기 위한 목적입니다.

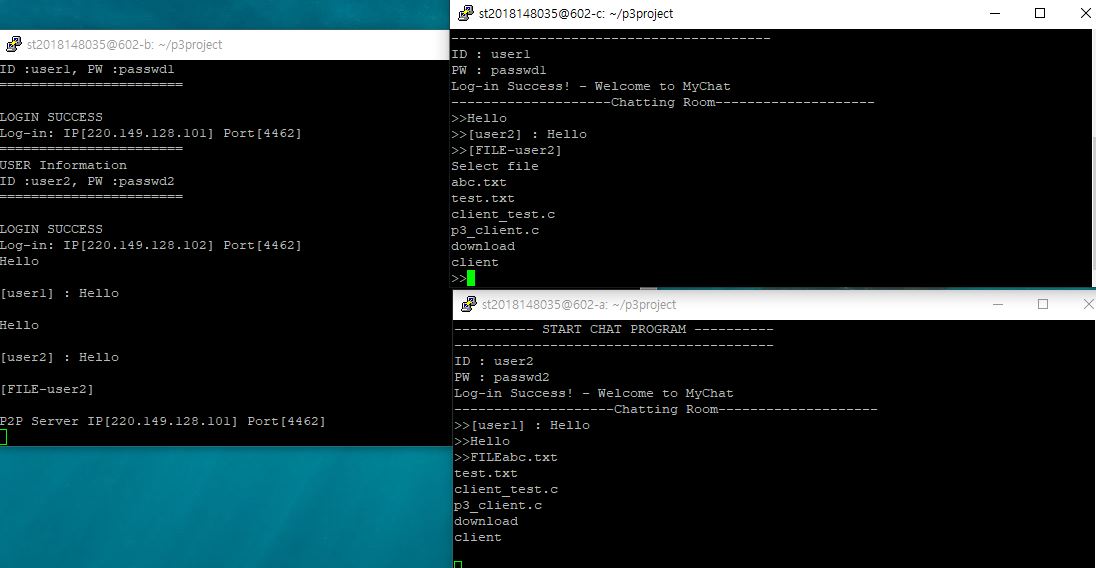


P2PInfo의 정보는 사용자가 서버에 접속하여 Login이 완료된 후 Login에 사용된 ID를 저장하고 IP와 Port 정보를 순서대로 수신합니다. 수신 시 recv() 함수를 이용하였으며 수신 버퍼를 P2PInfo 구조체에 바로 저장하였습니다. 저장 후 확인을 위해 저장된 정보를 printf()함수로 화면에 출력하여 확인 하였습니다.

2단계) P2P 요청 처리

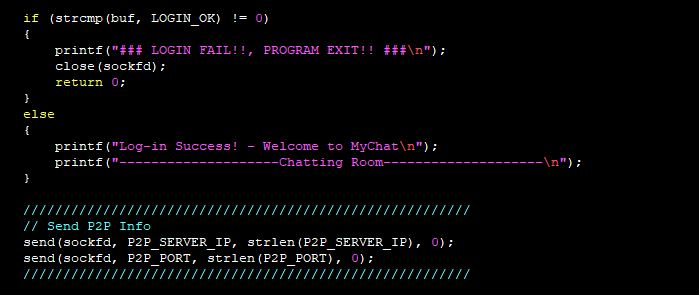


#Project3에서 파일 요청 방법은 “[FILE-user id]”입니다. #Project3은 특정 키워드를 확인하기 위해 특정 키워드 분석 로직을 추가하였습니다. 서버는 채팅 사용자의 대화 내용을 While Loop를 돌며 수신 합니다. 수신한 버퍼 내용을 확인하여 버퍼의 0번째부터 “[FILE-“문자열과 “[FILE-“문자열의 길이만큼 비교합니다. 만약 일치하지 않는 다면 수신된 내용을 #Project2과 같이 접속된 모든 사용자에게 송신합니다. 일치한다면 P2P 통신을 위한 준비를 합니다. 우선 수신된 “[FILE-user id]” 메시지에서 “user id”를 분리합니다. 분리된 “user id”를 이용하여 위에서 언급한 P2PInfo의 배열을 순회하여 id를 비교 매칭합니다. 매칭된 id의 클라이언트가 확인되면 그 클라이언트에게 “[FILE]”메시지를 전송합니다. 그 후 요청한 사용자의 P2PInfo의 IP와 Port 정보를 클라이언트에게 송신합니다.



위의 그림은 서버와 클라이언트들의 동작 화면입니다. 클라이언트들의 대화 메시지가 화면에 출력되며, “[FILE-user id]” 요청을 수신하면 서버는 P2P정보를 해당 클라이언트에게 전달합니다. P2P로 통신하는 내용들은 서버를 거치지 않기 때문에 출력되지 않습니다.

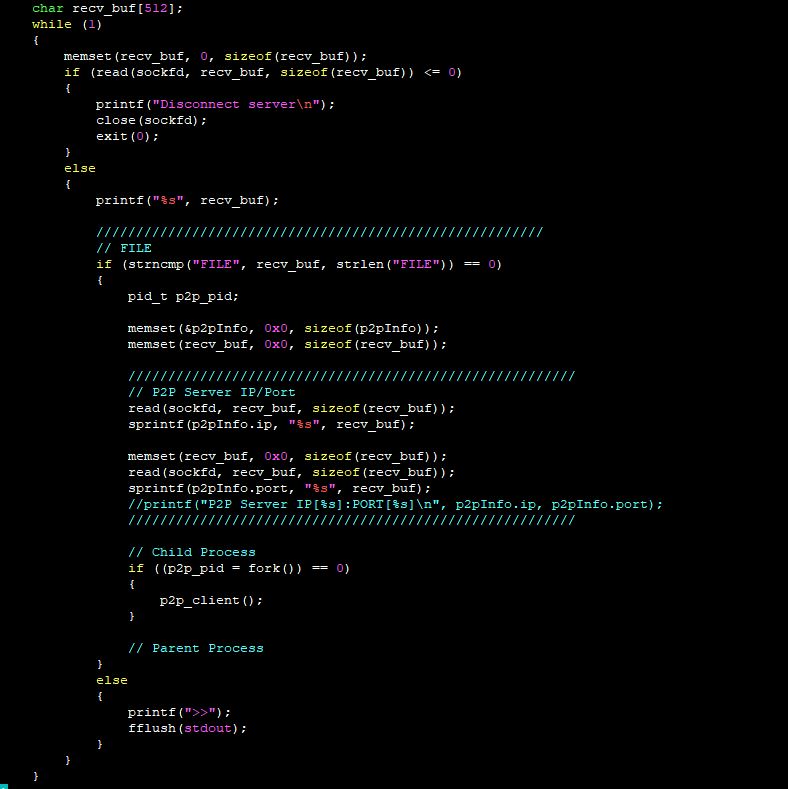
1단계) 클라이언트 - P2P 통신을 위한 정보 전송



#Project3에서 클라이언트가 Login을 성공하면 자신의 IP와 Port를 서버로 전송합니다. 채팅 서버를 거치지 않고 사용자들간의 P2P 통신을 하기 위한 정보입니다. P2P 통신에서 클라이언트 역할을 하면 이 정보를 가지고 P2P 서버에 접속을 시도합니다.



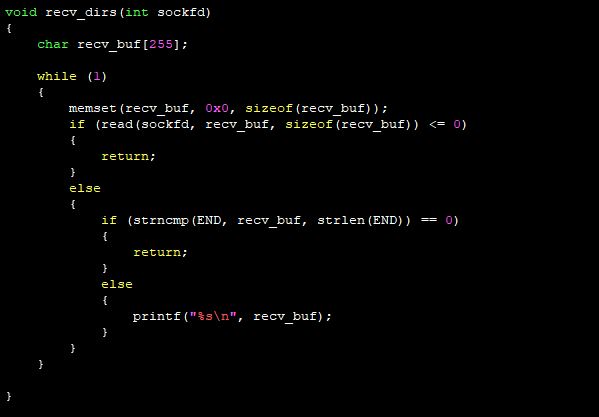
#Project3에서 클라이언트는 사용자로부터 입력받은 데이터를 확인합니다. 사용자의 입력 데이터가 “[FILE-“로 시작되는 문자열을 비교합니다. 문자열 비교가 일치하면 클라이언트는 fork()함수를 호출하여 Child Process를 생성합니다. Child Process의 역할은 TCP프로토콜을 사용하는 P2P 서버를 생성하기 위해서 입니다. Child Process는 시작할 때 p2p\_server()함수를 호출하여 소켓 생성 및 클라이언트의 접속을 대기하기 위한 준비 작업을 수행합니다. 자세한 내용은 아래에서 설명 드리겠습니다. Parent Process는 wait()함수를 이용하여 Child Process의 종료 할 때까지 대기합니다. Parent Process의 사용자 입력을 Child Process로 제어를 넘기기 위한 목적이기도 합니다. 만약, Parent Process가 Child Process의 종료를 기다리지 않는다면 Parent Process는 사용자의 입력을 받은 기능을 수행하기 때문에 P2P 서버에서 파일을 선택하기 위한 사용자 입력 정보가 서버로 전송되는 오류가 발생합니다.



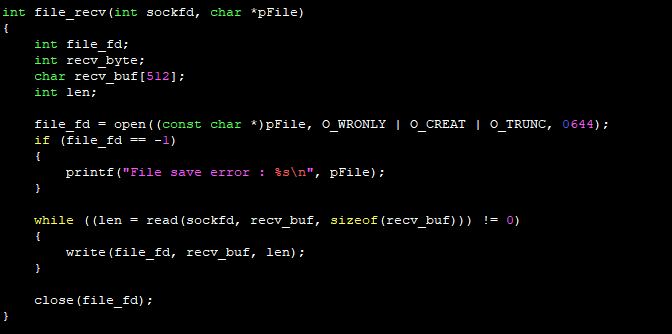
#Project3에서 서버로부터 수신되는 데이터도 확인합니다. 수신된 데이터가 “[FILE]”일 경우 클라이언트는 P2P 정보를 수신합니다. 추가로 수신하는 P2P 정보는 접속을 하기 위한 P2P 서버의 IP와 Port입니다. IP와 Port 수신이 완료되면 fork()를 호출하여 Child Process와 Parent Process로 프로세스를 분리합니다. Child Process는 P2P 클라이언트 역할을 담당하며, 수신된 IP와 Port를 이용하여 P2P 서버에 접속을 시도합니다. p2p\_client() 함수가 그 역할을 수행합니다. p2p\_client() 함수의 자세한 내용은 아래에 다시 설명 드리도록 하겠습니다. Parent Process는 다시 서버로부터 데이터를 수신하는 기능을 수행합니다.



위에서 간단히 설명 드린 p2p\_server()함수는 TCP프로토콜을 이용하여 Socket을 생성하고 서버 동작하는 설정 작업을 수행합니다. 그리고 accept() 함수를 호출하여 P2P 클라이언트가 접속하면 P2P 클라이언트와 통신을 수행하는 Socket을 전달 받아 P2P 클라이언트와 통신을 수행합니다. While Loop 돌며 P2P 클라이언트로부터 파일 리스트를 수신하기를 대기합니다. P2P 클라이언트로부터 파일 리스트를 수신을 받기 위해 recv\_dirs() 함수를 호출합니다. recv\_dirs() 함수는 P2P 클라이언트에서 송신되는 파일 리스트를 수신하는 작업을 수행합니다. 파일 리스트의 수신이 완료되면 화면에 보여줍니다. 사용자는 전송 받을 파일명을 입력하면 입력 받은 파일명은 P2P 클라이언트로 송신합니다. 그 후 파일 데이터 수신을 위해 file\_recv()함수를 호출하여 파일을 P2P 클라이언트로부터 수신하는 작업을 수행합니다. File\_recv() 함수가 return되면 생성된 P2P 서버 Socket과 accept() 함수로부터 전달받은 Socket을 close합니다. 그리고 프로세스를 종료합니다.



recv\_dirs()함수는 P2P 클라이언트로부터 파일 리스트를 수신합니다. 수신된 파일 리스트는 화면에현시되어 사용자가 파일 리스트 정보를 확인 할 수 있도록 합니다. 파일 리스트 송신 종료인 “END” 데이터가 수신되면 함수는 종료되어 호출한 곳으로 return하여 다음 작업을 수행합니다.

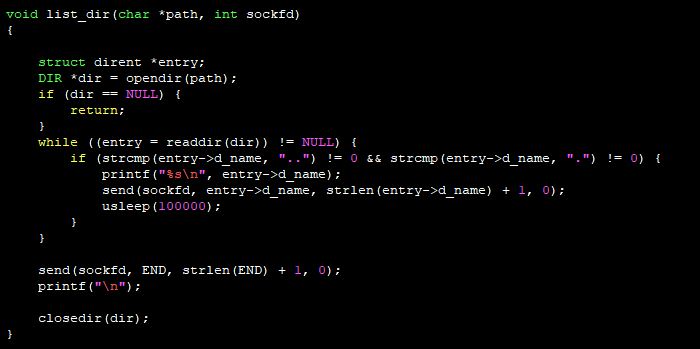


file\_recv()함수는 socket fd와 수신된 파일을 저장 할 경로를 인자로 받습니다. file\_recv() 함수는 전달받은 경로에 open() 함수를 이용하여 파일을 생성하고 file fd를 전달 받습니다. 생성한 파일의 권한을 “0644”로 설정하여 사용자는 읽기/쓰기 권한과 다른 사용자는 읽기 권한만 주어지게 설정하였습니다. 파일 데이터를 수신하면 생성한 파일에 수신된 데이터를 write합니다. 파일 데이터 수신이 완료되면 file fd를 close하고 호출한 곳으로 return하여 다음 작업을 수행합니다.

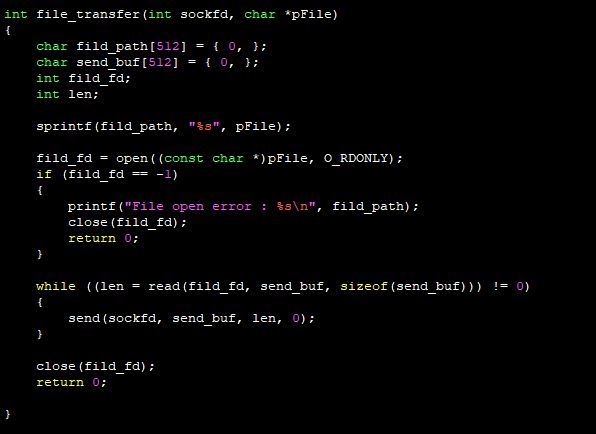


p2p\_client() 함수는 P2P 서버에 접속하기 위한 Socket을 생성하고 connect() 작업을 수행하는 client\_connect() 함수를 호출합니다. 호출이 완료되면 P2P 서버에 접속이 완료된 상태이면 생성된 Socket를 return 값으로 전달 받습니다. 그리고 getcwd() 함수를 호출하여 현재 디렉토리 위치 경로를 가져와 list\_dir()함수에 인자로 전달합니다. list\_dir()함수는 전달받은 경로의 파일 리스트 정보를 전달하는 작업을 수행합니다. list\_dir() 함수는 아래에서 다시 설명하겠습니다. List\_dir()함수가 작업을 완료하면 P2P 클라이언트는 전송 할 파일명 수신을 기다립니다. 파일명이 수신되면 file\_transfer() 함수를 호출하여 파일 전송 작업을 수행합니다. 전송이 완료되면 “>>File transfer Success!!”문구가 출력되고 Socket을 close합니다. 그리고 Process는 종료됩니다.

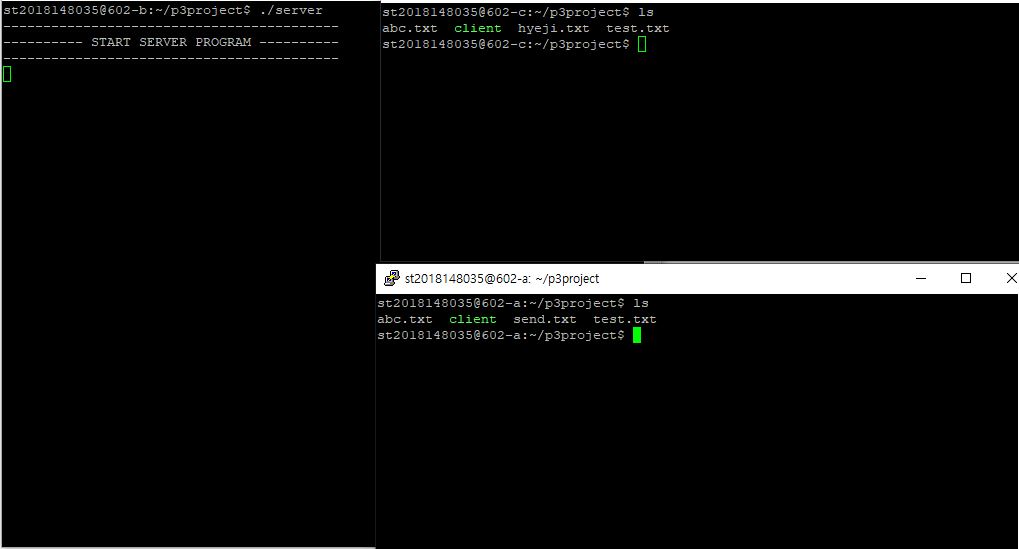
만약, client\_connect() 함수로부터 전달받은 Socket 값이 정상이 아니면 P2P 서버와의 접속은 실패입니다. P2P 서버의 접속이 성공하지 못하면 함수를 종료하여 P2P Client 프로세스를 종료합니다.



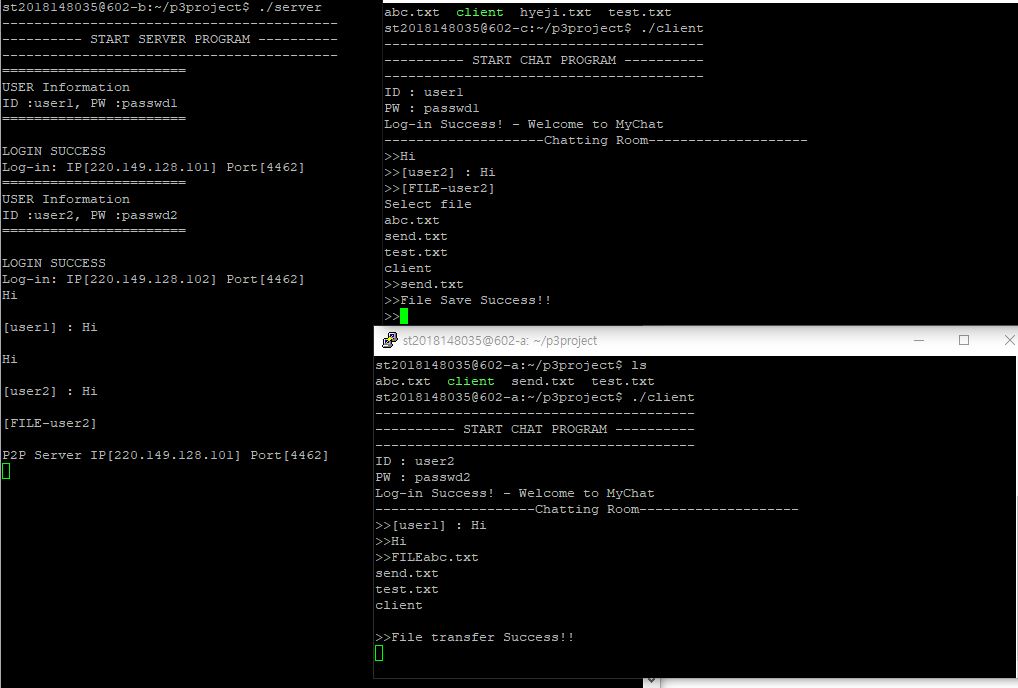
list\_dir() 함수는 전달받은 경로의 디렉토리를 opendir() 함수를 호출하여 디렉토리 스트림 포인트를 전달 받습니다. 이 포인트를 readdir() 함수의 인자로 전달하여 파일의 정보가 담긴 struct dirent 포인터를 전달 받아 while Loop를 돌며 파일정보를 P2P 서버로 송신합니다. 파일의 이름중 “.”과 “..”의 현재 디렉토리와 상위 디렉토리를 가리기는 파일이기 때문에 송신을 생략합니다. 모든 파일을 정보를 송신하면 마지막으로 “END” 데이터를 송신하여 파일 리스트 송신 작업을 마무리합니다. 열려 있는 디렉토리 스트림 포인터는 closedir()함수를 호출하여 close한 후 return하여 함수를 종료합니다. 그리고 list\_dir() 함수를 호출한 곳으로 돌아가 다음 작업을 수행합니다.



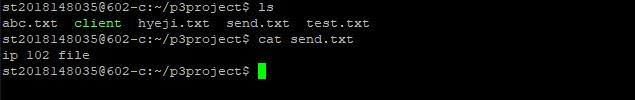
file\_transfer()함수는 P2P 서버로부터 송신한 파일명을 인수로 전달 받습니다. 전달 받은 파일명의 파일을 open()함수를 이용하여 읽기 모드로 파일 열기를 수행합니다. 파일이 정상적으로 열리면 file fd를 전달 받습니다. 전달 받은 file fd를 이용하여 read()함수를 호출하여 파일의 데이터를 순차적으로 send buffer로 읽어 들입니다. 그리고 P2P 서버로 송신합니다. 모든 파일 데이터가 P2P 서버로 송신이 완료되면 file fd를 close하여 파일을 닫습니다. 그리고 return하여 file\_transfer() 함수를 호출한 곳으로 돌아가 다음 작업을 수행합니다.



채팅 서버에 접속하기 전 클라이언트의 파일 리스트 입니다. 위의 사용자는 “220.149.128.101”IP이며, 두번째 사용자는 “220.149.128.102”IP 입니다.



접속 후 클라이언트들의 인사 메시지를 주고 받습니다. 그리고 첫번째 사용자의 요청으로 P2P 통신이 이루어 집니다. 첫번째 사용자의 “send.txt” 파일 전송 요청으로 두번째 사용자는 “send.txt” 파일을 전송합니다. 전송이 완료되면 결과를 화면에 현시하고 P2P 파일 전송 통신을 종료됩니다. 채팅 서버는 클라이언트들간의 메시지만 현시하고 파일 전송에 대한 내용은 채팅 서버를 거치지 않기 때문에 보여지지 않습니다.



전송된 “send.txt”파일을 확인 합니다. 첫번째 사용자의 디렉토리에 “send.txt” 파일을 ls 명령으로 확인 하였으면, cat 명령을 이용하여 파일의 내용도 확인 하였습니다.