임베디드 운영체제

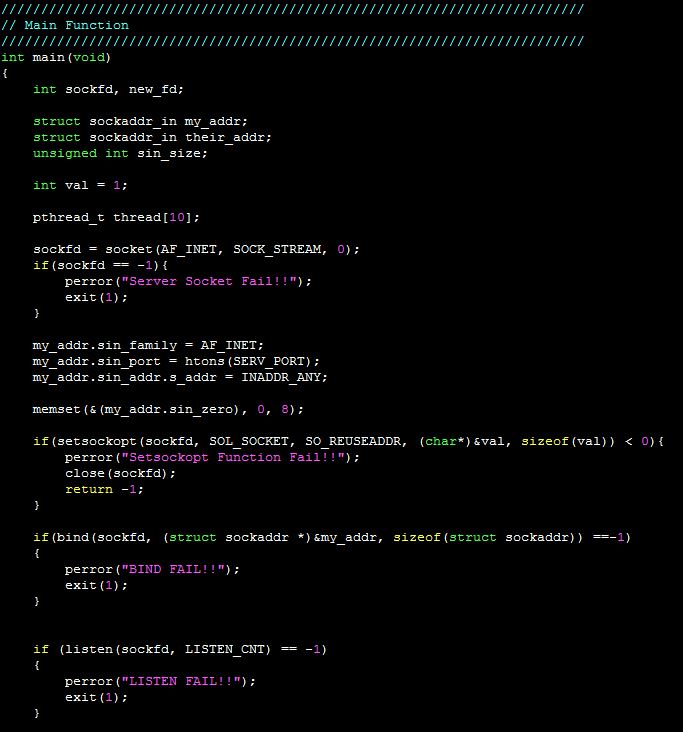
Final Project #2

전자공학부 임베디드시스템 전공

2018148035 한혜지

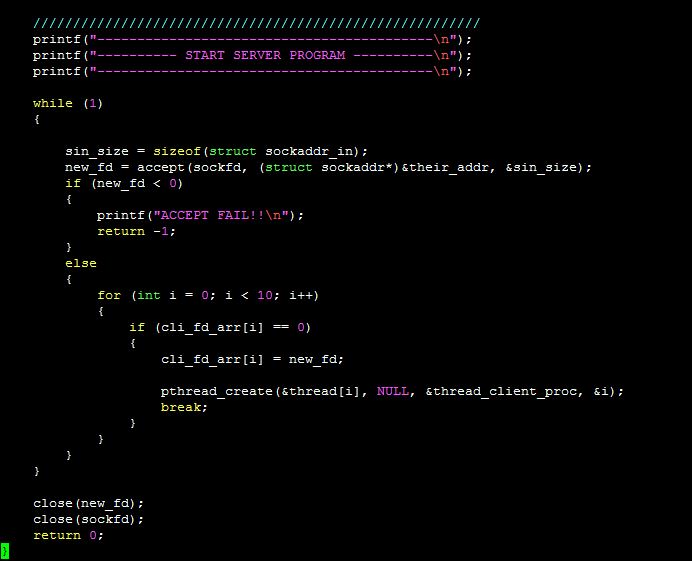
2018146036 황서영

1단계) 서버 프로그램.



서버 프로그램 실행 시 제일 처음 호출되는 main()함수에서 TCP프로토콜을 사용하는 Server Socket을 생성합니다. Server Socket 생성이 비정상일 경우 에러 메시지를 현시하고 프로그램을 종료합니다. SO\_REUSEADDR Socket option을 설정하여 이미 사용된 주소를 재사용 가능하도록 설정하였으며, bind() 함수를 호출하여 Socket에 설정된 정보를 설정하고 운영체제에 등록하도록 하였습니다.

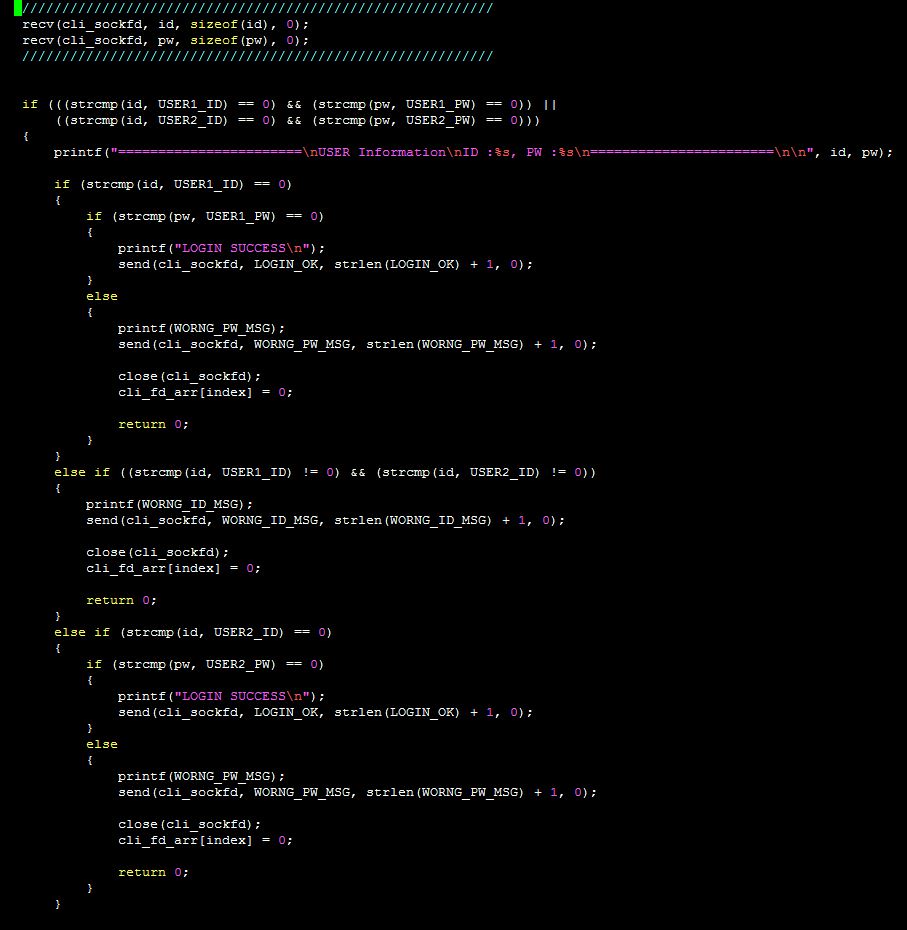
마지막으로 listen()함수로 클라이언트 접속 요청 대기합니다.



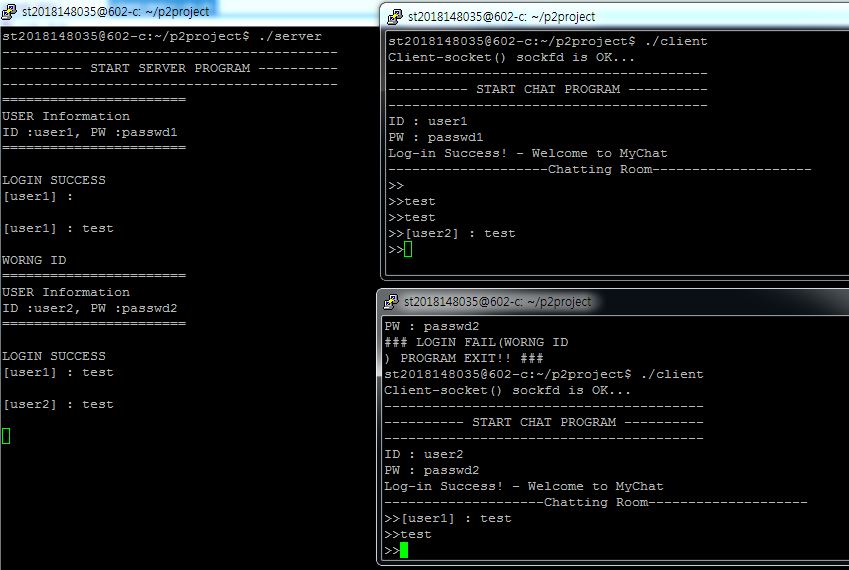
클라이언트에서 접속 요청을 하면 accept()함수를 통하여 클라이언트와 통산하기 위한 Socket을 커널에서 생성하여 전달합니다. 이 Socket을 주로 Client Socket이라고 부릅니다. Client Socket은 전역 배열로 관리하며 수신된 메시지를 전체에 송신 할 때 사용됩니다. 그리고 accept() 함수는 접속한 클라이언트의 정보를 구하기 위한 변수를 인자로 받고 있습니다. 클라이언트가 정상적으로 접속하면 접속한 클라이언트의 정보를 확인 할 수 있습니다.

클라이언트와 통신을 처리하기 위해 Thread를 사용하였습니다. Thread는 최대 10개까지 생성이 가능합니다.

2단계) Thread – Login

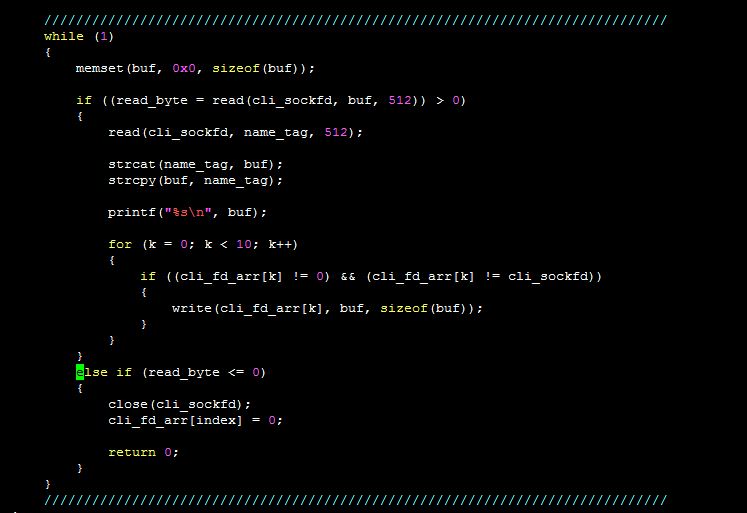


클라이언트가 서버에 접속을 하며 서버는 클라이언트와의 통신을 담당하는 Thread를 생성합니다. 위의 소스내용은 클라이언트와 통신을 담당하는 Thread의 내용입니다. 우선 클라이언트로부터 Login 정보(ID/Passwd)를 수신합니다. 수신된 Login 정보가 정상 또는 비정상인지 확인 합니다. 정상이면 클라이언트에게 정상 메시지를 전송하며, Login에 성공하지 못하였을 경우는 실패 원인에 대한 간단한 메시지가 전송됩니다. Login 실패 시 클라이언트를 close하고 Thread로 종료합니다.



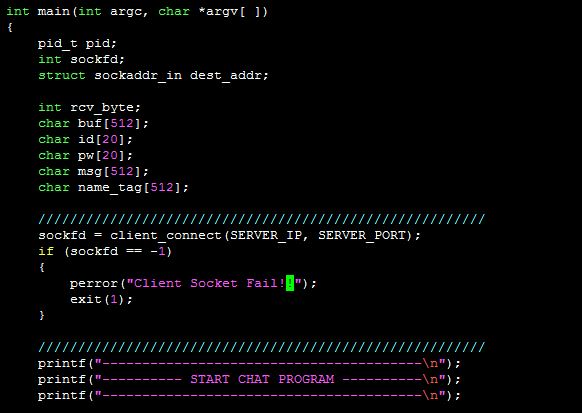
서버 실행 및 로그인 처리 화면입니다. 서버를 실행하면 프로그램 시작 문구가 현시된 후 클라이언트의 접속을 대기하고 있습니다. #user1 사용자가 Login을 시도하여 정상 되었으며, #user2 사용자는 처음 시도한 Login은 실패하여 다시 시도하였고, 두 번째 시도는 Login이 정상 처리되는 과정입니다.

2단계) Thread – 송/수신 처리

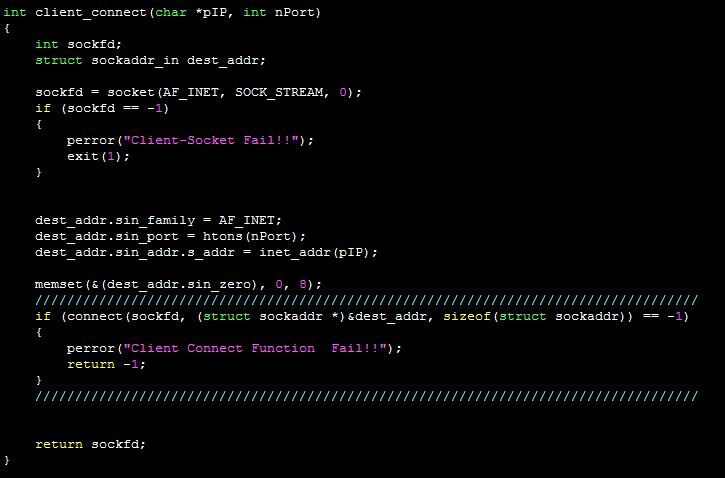


클라이언트로부터 채팅 메시지와 ID를 수신하면 strcat() 함수를 이용하여 “[ID] : 메시지” 형태로 송신 데이터를 생성합니다. 그리고 접속된 클라이언트 모두에게 전송합니다. 만약 클라이언트 메시지를 수신하는 read()가 비정상 반환값을 전달하면 클라이언트와의 접속이 종료된 것으로 판단하여 클라이언트 Socket을 close하고 Thread를 종료합니다.

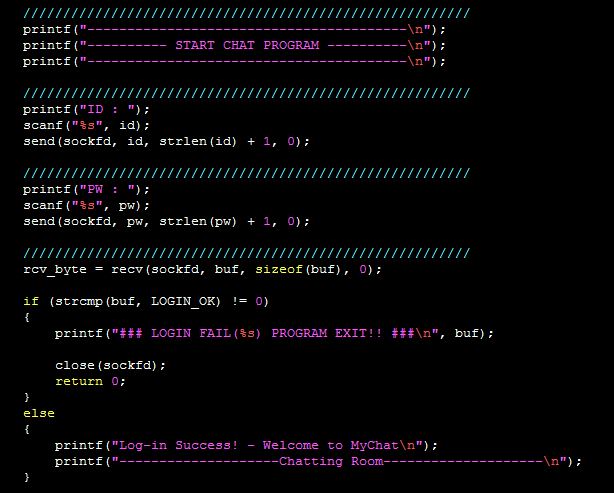
3단계) 클라이언트



클라이언트의 메인 함수에서는 서버 접속을 위한 connect()를 담당하는 함수를 별도로 만들었습니다. 그 함수는 client\_connect()이며, 서버의 IP와 Port를 인자로 받아 서버에 접속을 시도하고 결과를 반환합니다. 함수를 분리한 이유는 #Project3(P2P Client)에서도 사용하기 위해서 입니다.



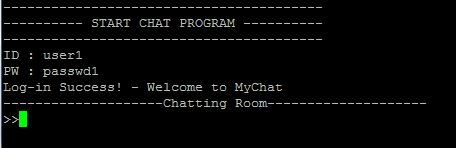
client\_connect()함수는 TCP socket을 생성하고, 주소정보에 전달받은 서버 정보를 설정하여 connect()함수를 이용하여 서버에 접속을 시도합니다.



서버와 접속이 완료되면 프로그램 시작 문구를 보여줍니다. 그 다음 Login을 위한 사용자 정보를 scanf()함수를 통해 입력 받습니다. 입력된 사용자 정보를 서버로 송신하여 Login을 결과를 recv()함수를 이용하여 수신대기 합니다. 수신이 완료되면 수신된 메시지를 문자열 비교합니다. 수신된 문구가 “LOGIN\_OK”와 일치하면 Login 성공 문구를 보여줍니다. Login에 실패 할 경우 sockfd를 close하고 return하여 프로그램을 종료합니다.



클라이언트는 Login이 성공하면 fork()함수를 이용하여 2개의 Process로 분리됩니다. Child Process는 데이터 수신을 담당하고, Parent Process는 표준 입력 처리 및 입력된 데이터 송신을 담당합니다. Child Process는 fork() 후 While Loop를 돌며 read()를 이용하여 서버로부터 데이터가 수신되기를 기다립니다. 데이터가 수신되면 화면에 현시하고 다시 데이터를 기다립니다. 만약, read() 함수에서 반환된 값이 0 또는 0보다 작으면 Socket이 종료된 것으로 판단하여 Child Process를 종료합니다. Parent Process도 While Loop를 돌며 사용자로부터의 입력을 대기합니다. 입력이 완료되면 입력된 데이터를 write() 함수를 이용하여 서버로 전송합니다.



클라이언트 실행 및 사용자 입력 대기 화면입니다.

아래의 화면은 #user1 사용자와 #user2 사용자의 채팅중인 화면입니다.

