

<임베디드 운영체제>

김수민 교수님

Final Project #3 보고서

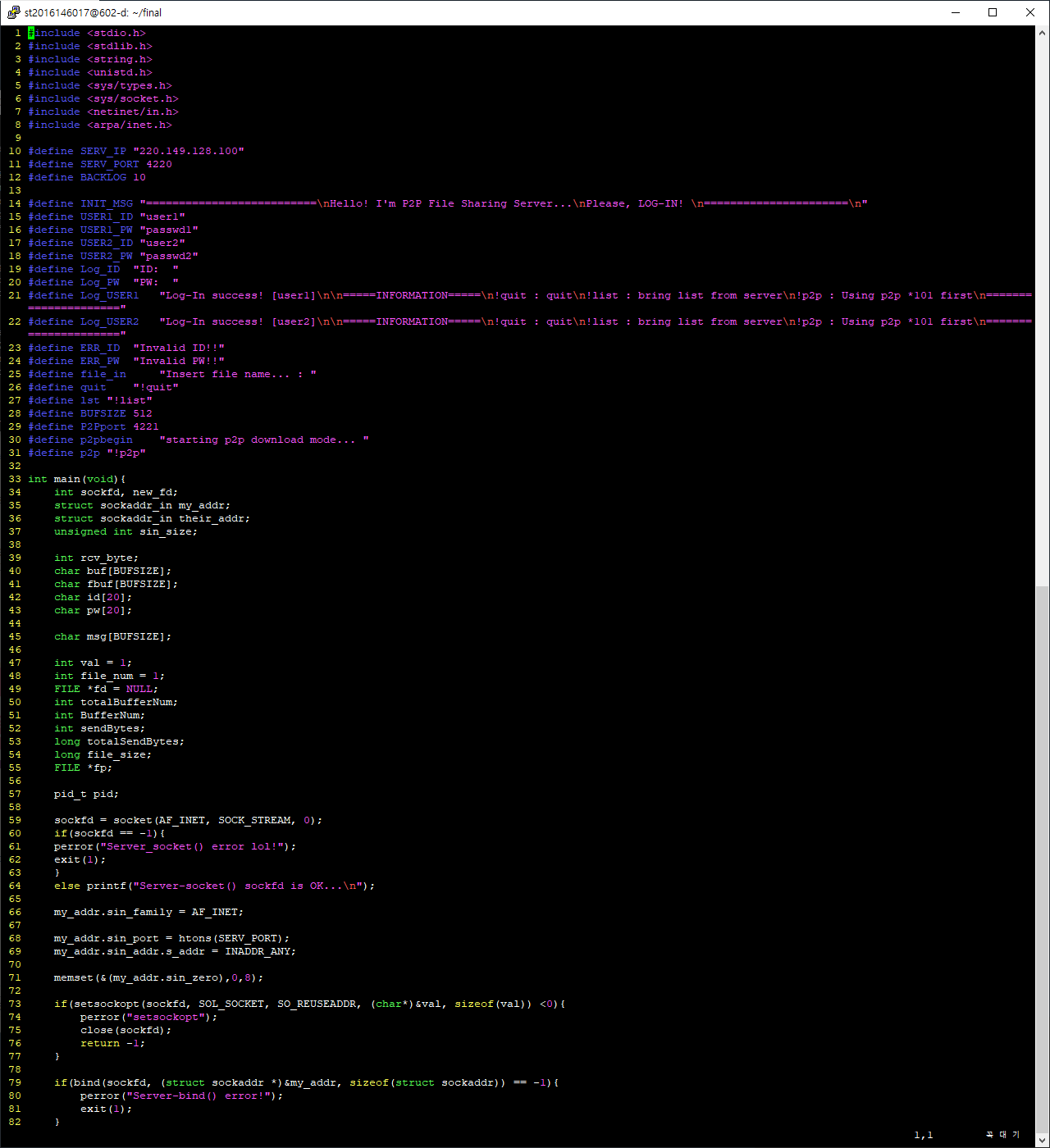
6팀

2016146006 김석현

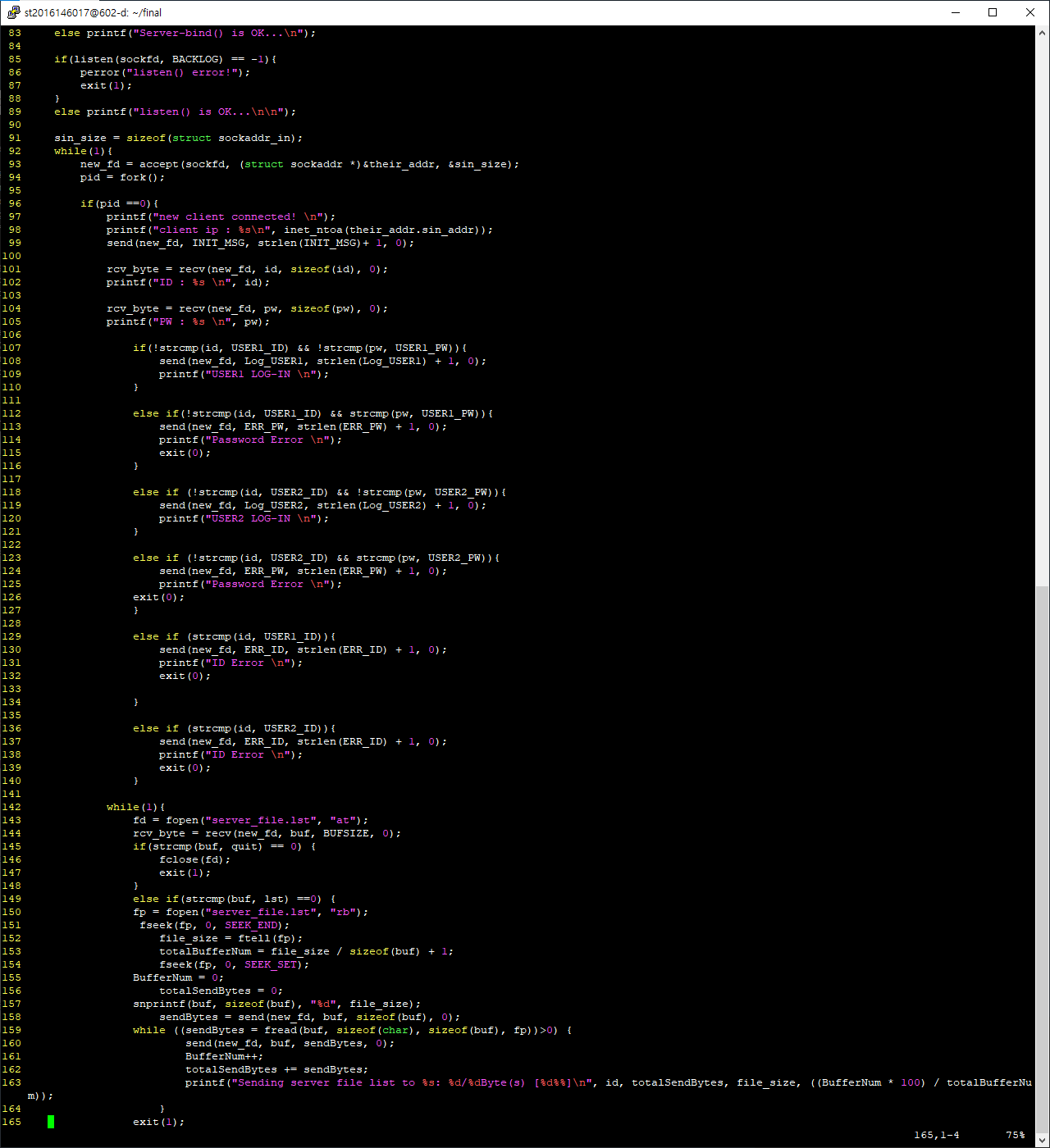
2016146017 김찬희

서버측(100) 코드의 캡처본

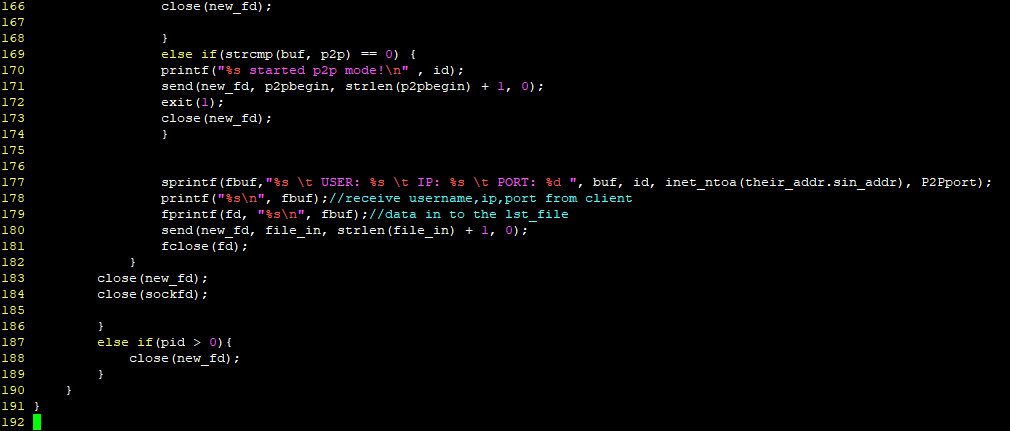
Line 1~82



Line 83~165

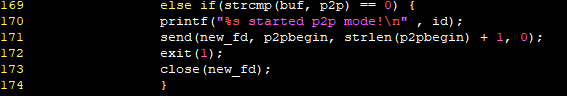


Line 166~



기존 project #2 와 달라진 점은 로그인 성공 시 명령어와 명령어의 동작을 Log\_USER1, Log\_USER2에 집어넣고 전송하여 사용자에게 어떤 명령어가 있는지 알려주도록 했습니다.



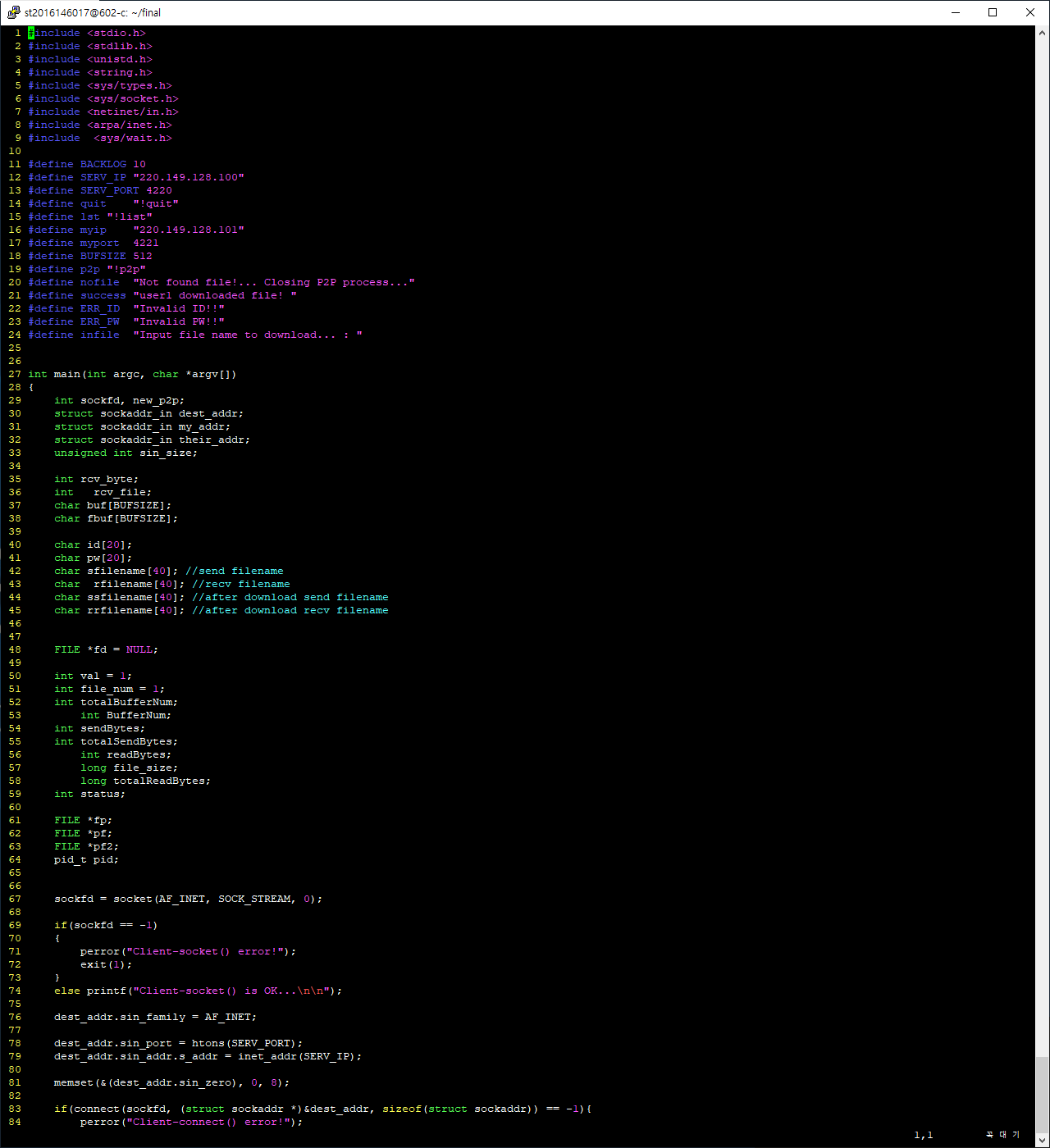


사용자가 로그인 후 !p2p 명령어를 입력하게 되면 명령어가 buf로 들어가고, 미리 define 된 p2p라는 변수와 strcmp 함수를 통해 비교를 하여 맞다면 어떤 user가 p2p 모드를 시작했는지 서버측에 출력해준 뒤, p2p 모드를 시작한다는 안내를 new\_fd 소켓으로 전송 후 자식 프로세스를 종료, new\_fd 연결을 끊어줍니다.



클라이언트(101) 코드

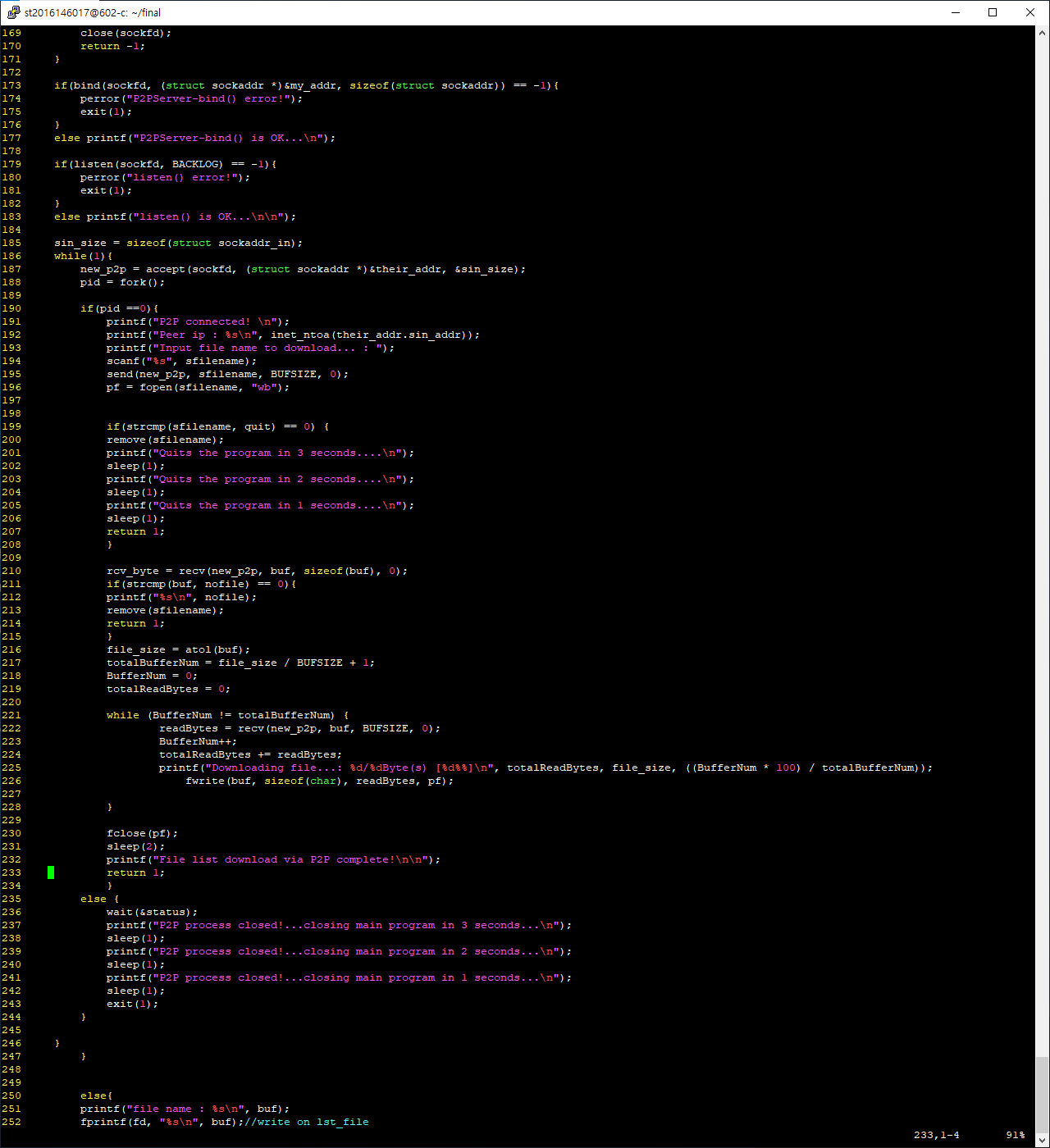
Line 1~84



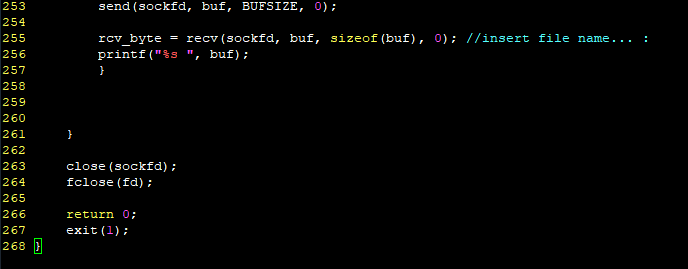
Line 85~168



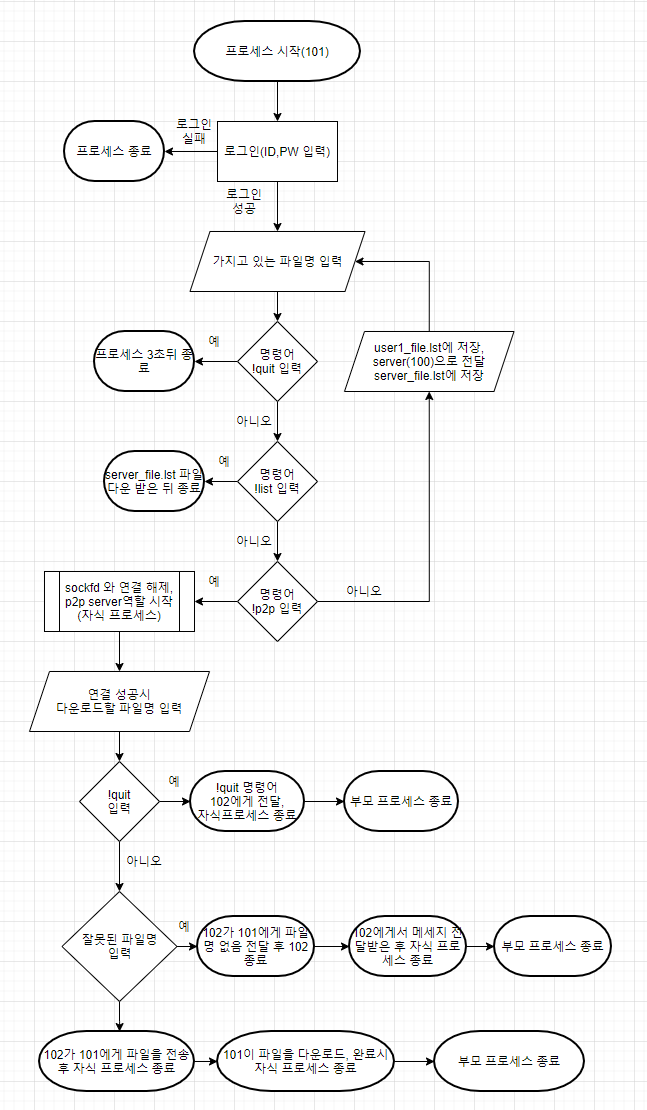
Line 169~252



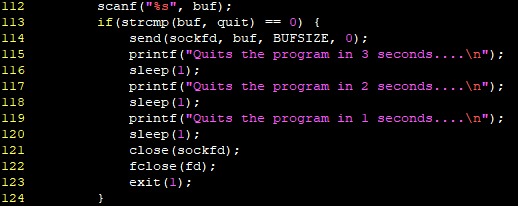
Line 253~268

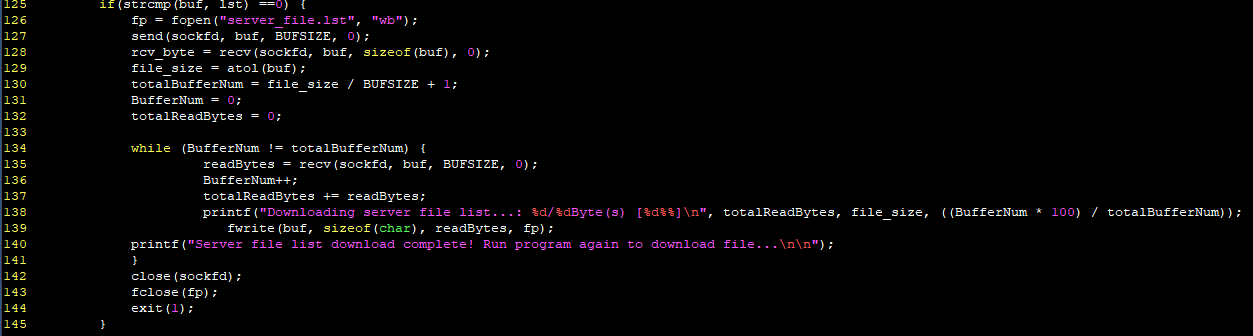


101의 프로세스 순서도

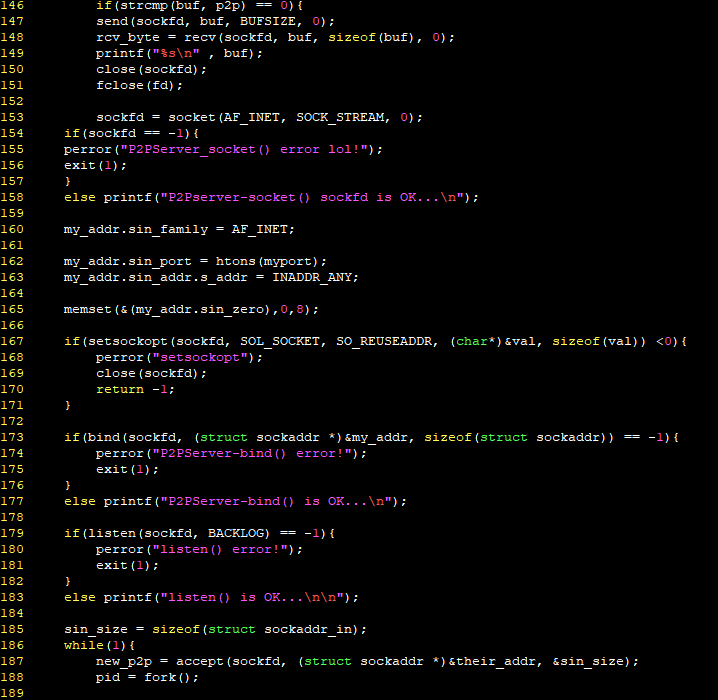


101에서 project#2는 명령어를 !quit, !list 두개로만 설정하여 리스트 파일을 작성하다가 작성을 끝내고 싶은 경우에는 !quit 명령어를 통해 프로세스를 종료시켰으며,



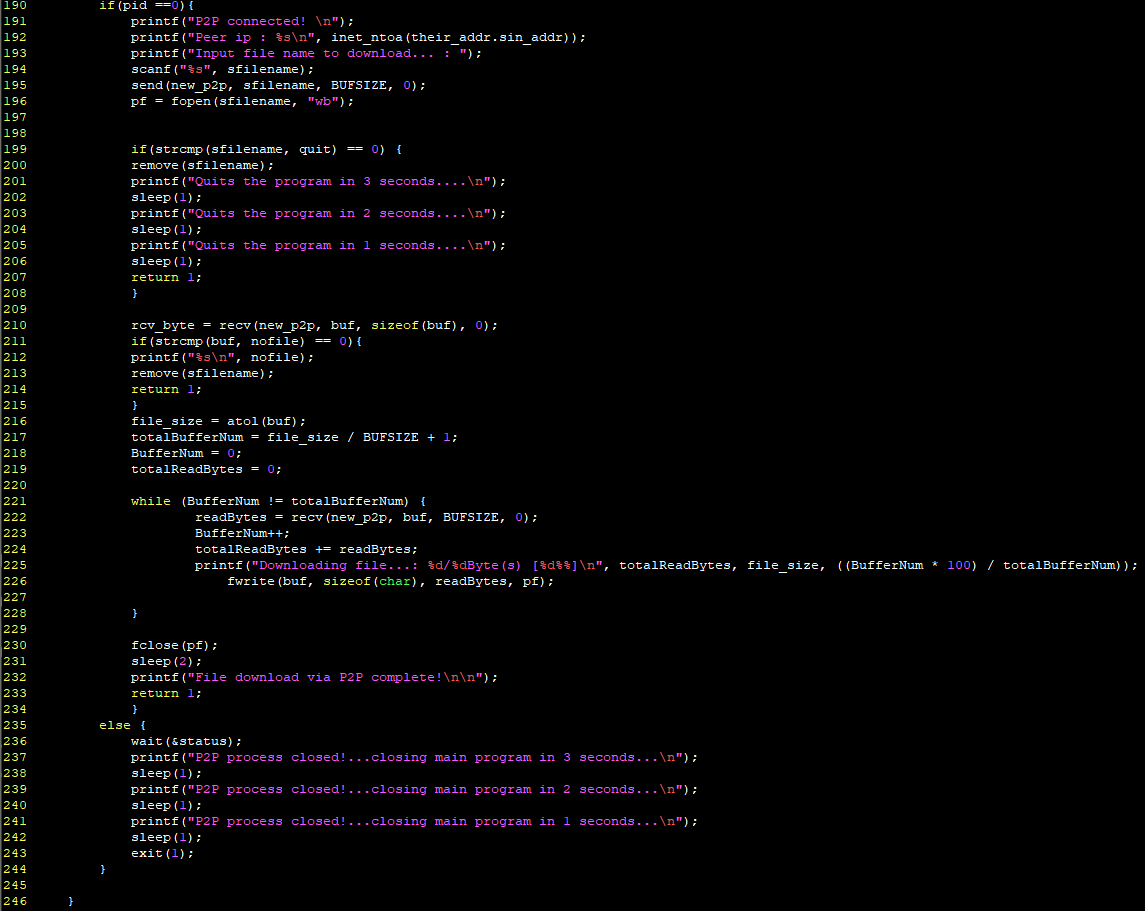
서버로부터 리스트 파일을 받아오고 싶은 경우에는 !list 명령어로 server\_file.lst 파일을 받아온 뒤 프로세스를 종료시켰습니다. 

여기에 이어서 !p2p 명령어를 추가한 뒤, buf를 sockfd에 전송하고 p2p 모드에 진입했음을 printf시켜주고 sockfd의 연결을 끊고 p2p 서버의 역할을 위해 server.c 예제의 socket 생성 코드를 추가했습니다.



while(1) 무한 루프를 통해 지속적으로 클라이언트를 accept 하여 밑에 있는 fork함수가 무한히 동작하지 않게 막았습니다.

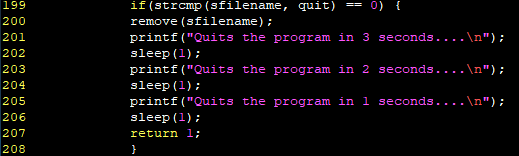
자식 프로세스 부분의 코드:



Accept를 통해 연결에 성공하면 fork 함수로 자식 프로세스를 생성합니다. 자식프로세스에서는 P2P가 연결되었다는 문구를 printf 하면서 inet\_ntoa(their\_addr.sin\_addr)로 peer의 ip를 표시해줬습니다. (line 191~192).

파일 이름을 입력하라는 문구가 뜨면 사용자는 다운로드를 원하는 파일을 받아두었던 server\_file.lst 파일을 참고, 파일명을 입력합니다.

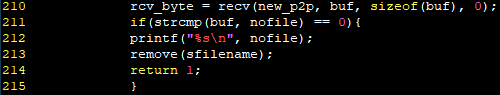
입력한 파일명이 !quit 명령어일 경우, 부모 프로세스에서 적용하였던 코드와 같이 3초뒤 자식프로세스가 꺼지게 되고, 236번째줄의 wait함수가 동작, 부모프로세스도 꺼지도록 하였습니다.



Wait함수를 사용하기 위해 추가적으로 <sys/wait.h> 헤더 파일을 추가했습니다.

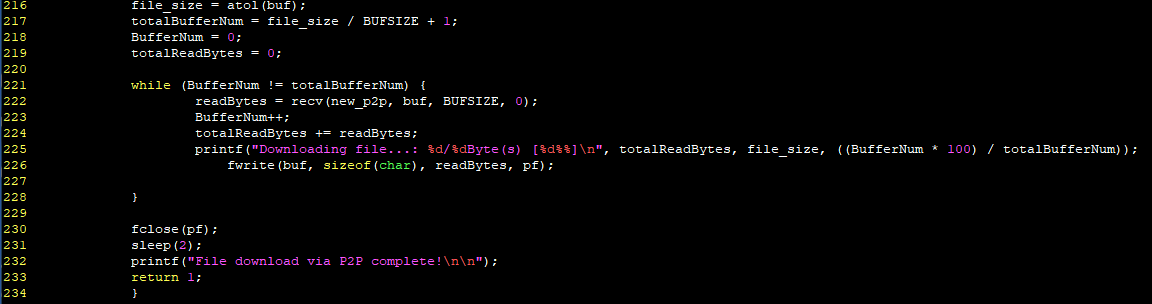


입력한 파일명이 부정확한 경우: 102 p2p client가 파일명이 없다는 메시지를 101에게 send하게되고 101은 recv한 buf값을 nofile과 strcmp로 비교, 파일이 없음을 출력하고 자식, 부모 프로세스를 종료시켰습니다.



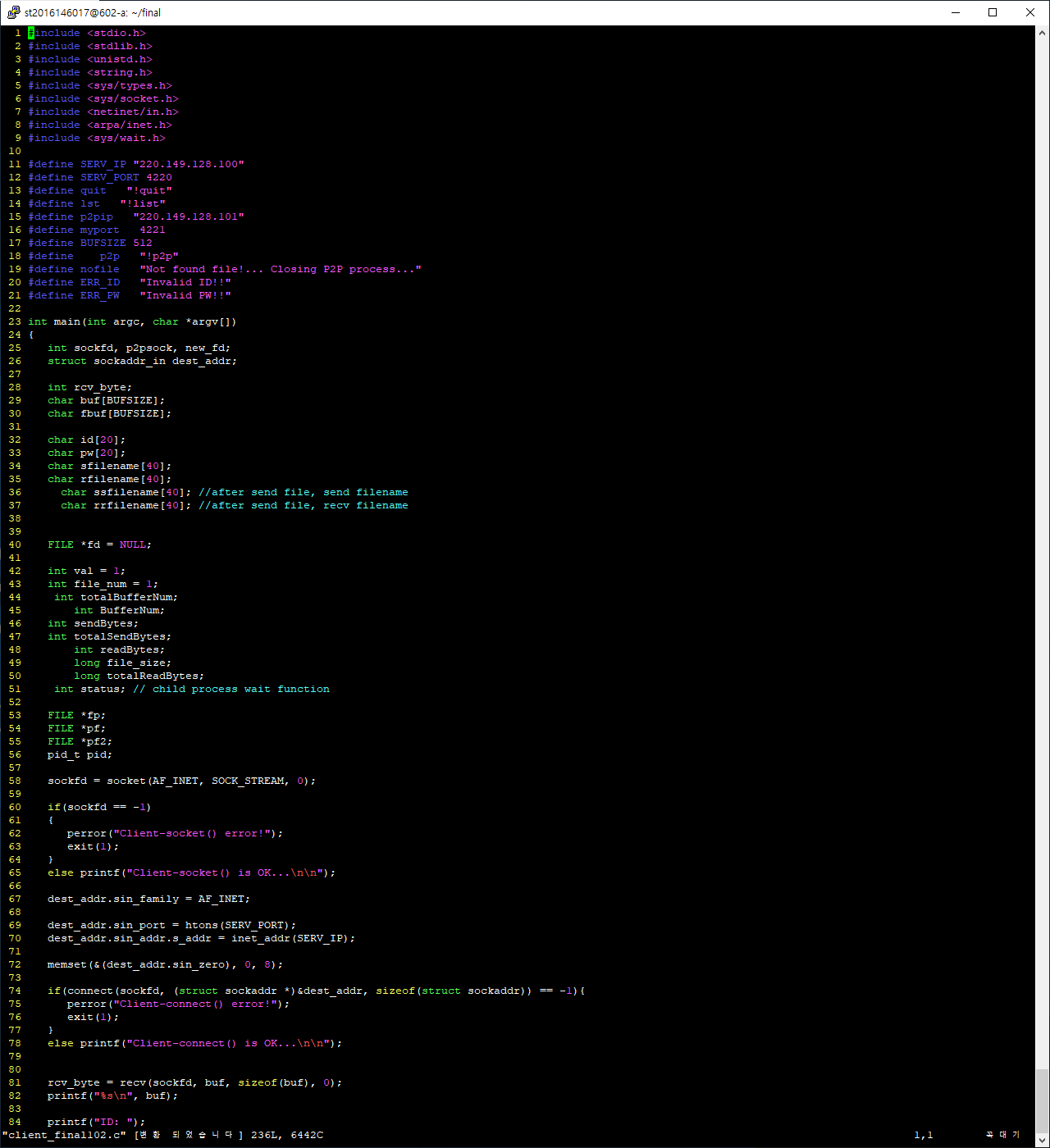
101 코드에서 입력한 파일명을 sfilename에 저장하고 fopen함수를 통해 “wb”옵션으로 파일을 생성하였으므로 !quit 명령어를 입력했을 때와 파일명이 부정확한 경우에는 remove(sfilename)으로 파일이 생성되지 않도록 작성했습니다.

올바른 파일명을 입력했을 때에는 102로부터 파일을 전송받으며 몇 바이트 중 몇 바이트를 다운로드 했는지 안내 문구가 뜨도록 했습니다. 파일을 모두 다운로드하면 완료했다는 문구가 뜨고 자식, 부모 프로세스를 종료시킵니다.



클라이언트(102) 코드

Line 1~84



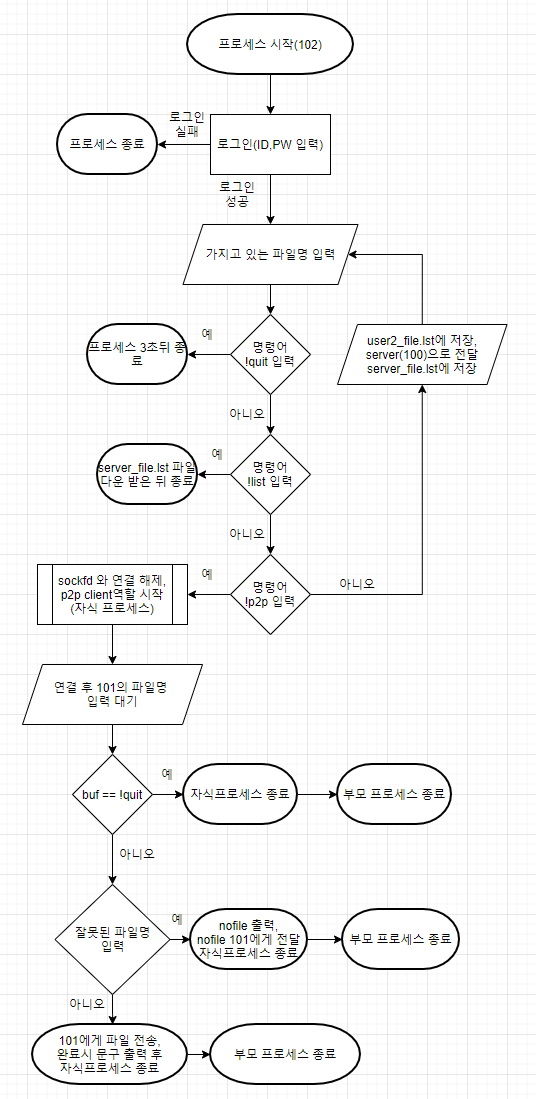
Line 85~168



Line 169~236



클라이언트 102의 순서도

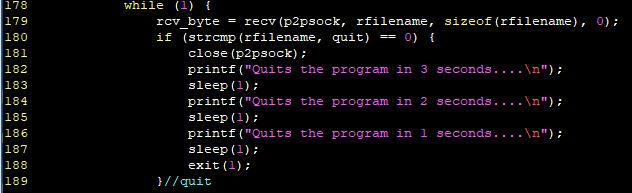


!p2p 명령어를 받으면 sockfd와 연결을 끊고 break문으로 while문을 벗어난 후 바로 fork를 진행, 자식 프로세스를 생성했습니다.

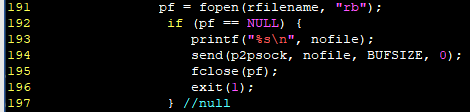
!p2p 입력을 받기 전까진 101의 동작과 같으나 101은 p2p에서 서버를 담당하지만 102는 client를 담당하기에 코드가 달라지게 되었습니다. Client.c의 코드를 참고했으며 포트번호와 ip를 101에 맞춰주었습니다.



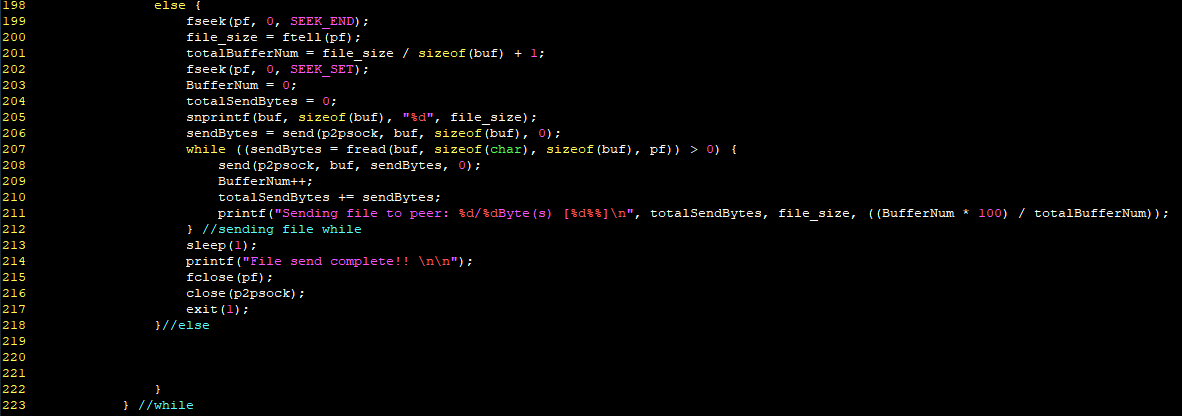
101에서 전송할 파일명을 recv 함수를 통해 rfilename 변수에 저장, !quit 명령어와 같은지 비교합니다.



명령어가 아니라면 fopen함수로 rfilename을 “rb” 옵션으로 열게 되며 파일명이 잘못되었을 경우 102에서 파일이 없음을 출력하고 nofile을 p2psock(101)로 전송하고 자식, 부모 프로세스를 종료합니다.

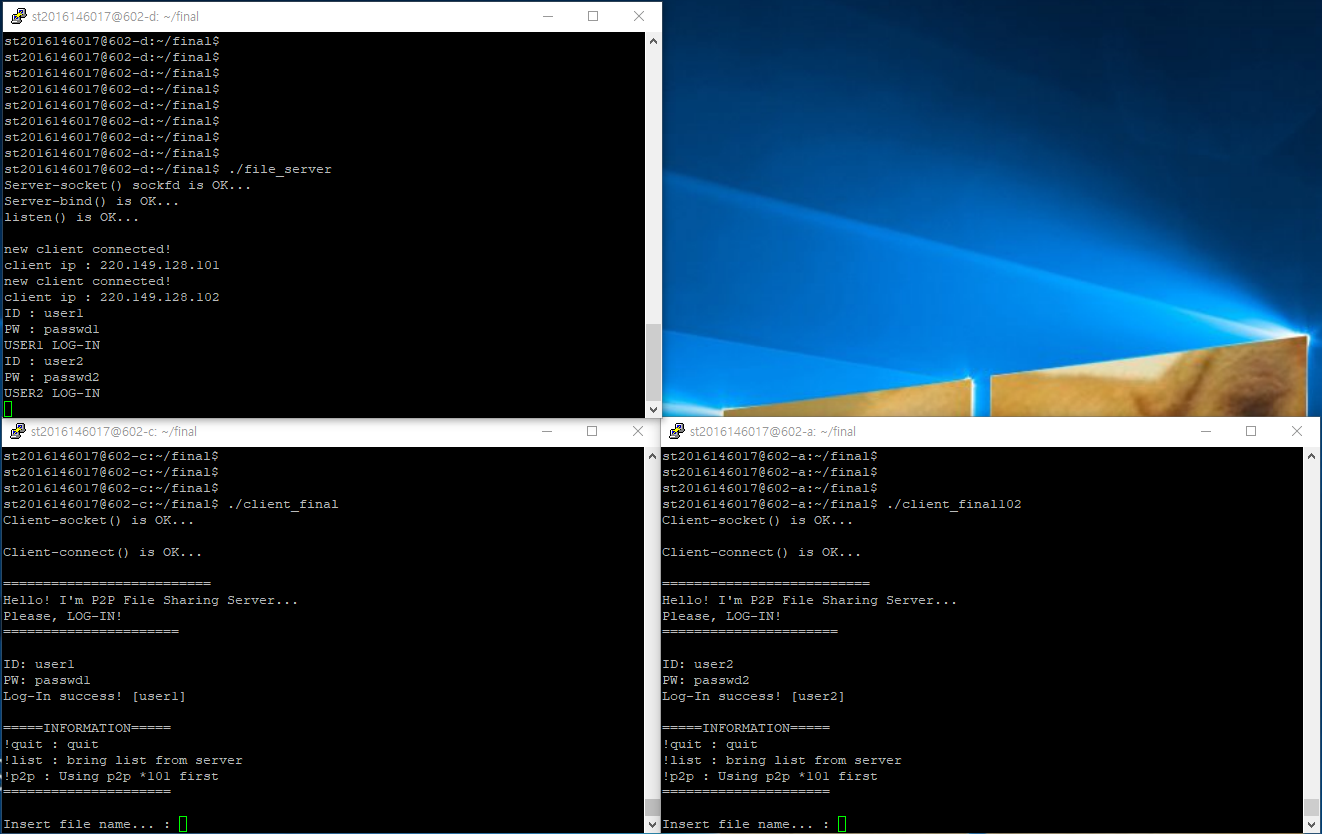


올바른 파일명을 입력하면 project#2의 list파일 전송과 같은 방식으로 peer에게 얼마만큼의 데이터를 전송했는지 출력하고 출력이 끝나면 파일 전송이 끝났음을 출력 후 자식, 부모 프로세스를 종료합니다.



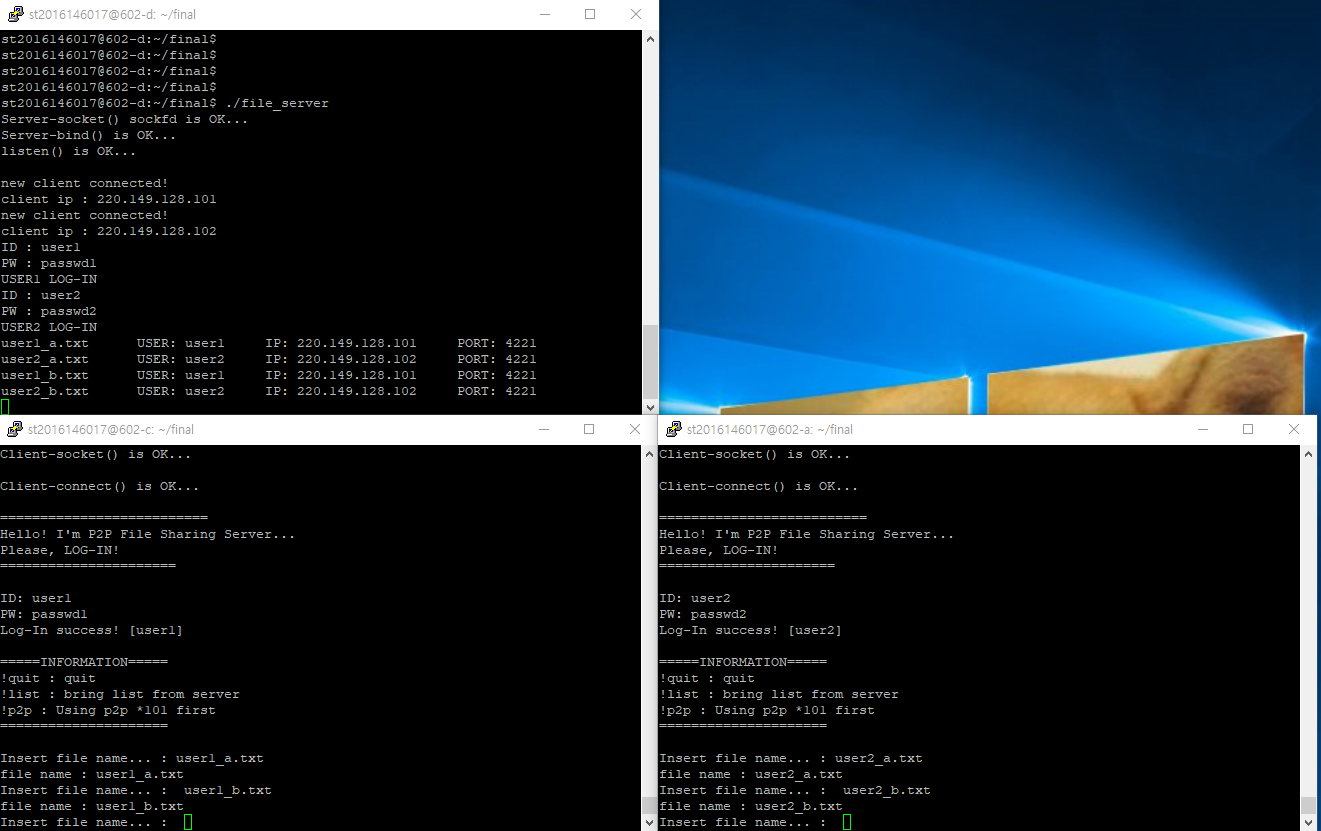
코드 동작 화면:

(로그인)



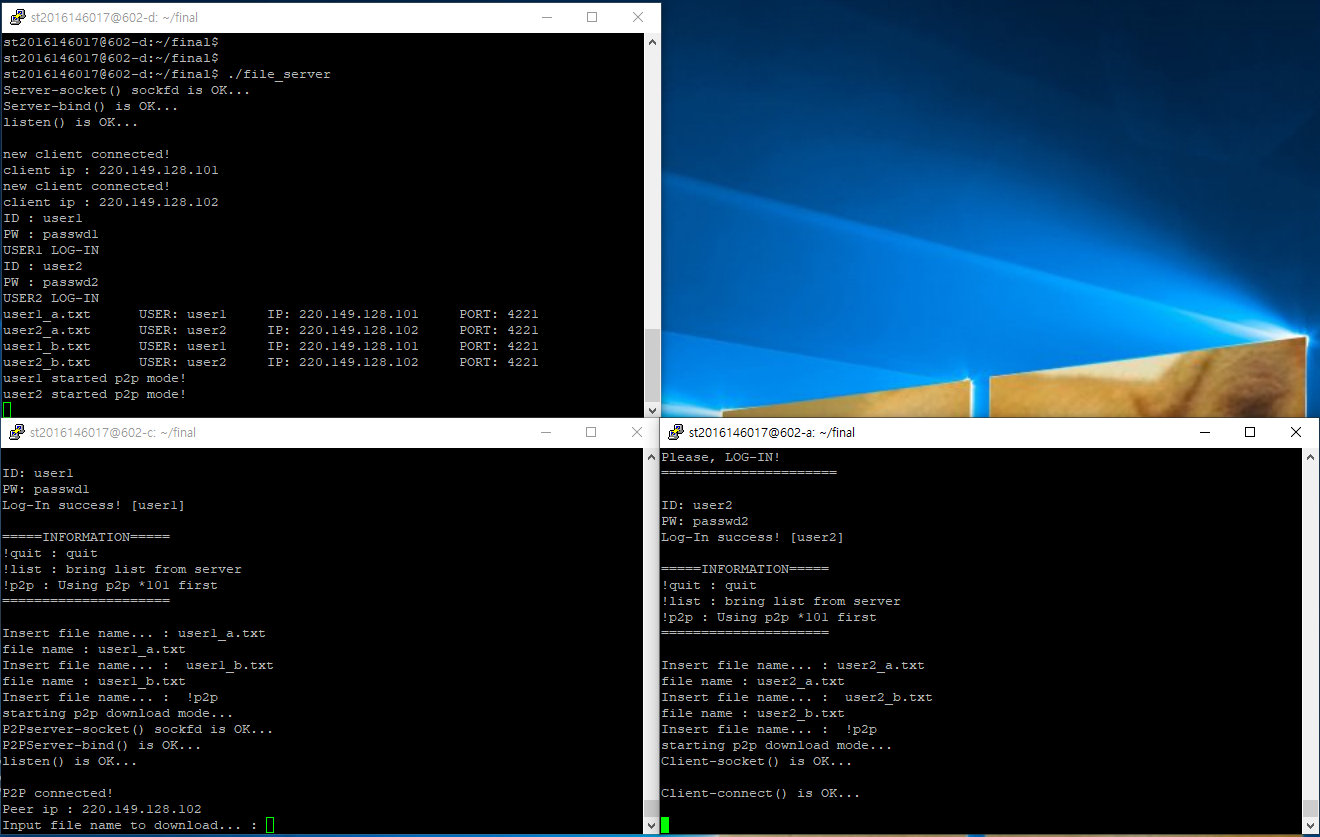
서버는 client의 ip를 제대로 가져오고 로그인도 정상적으로 작동함.

(파일 리스트 작성)



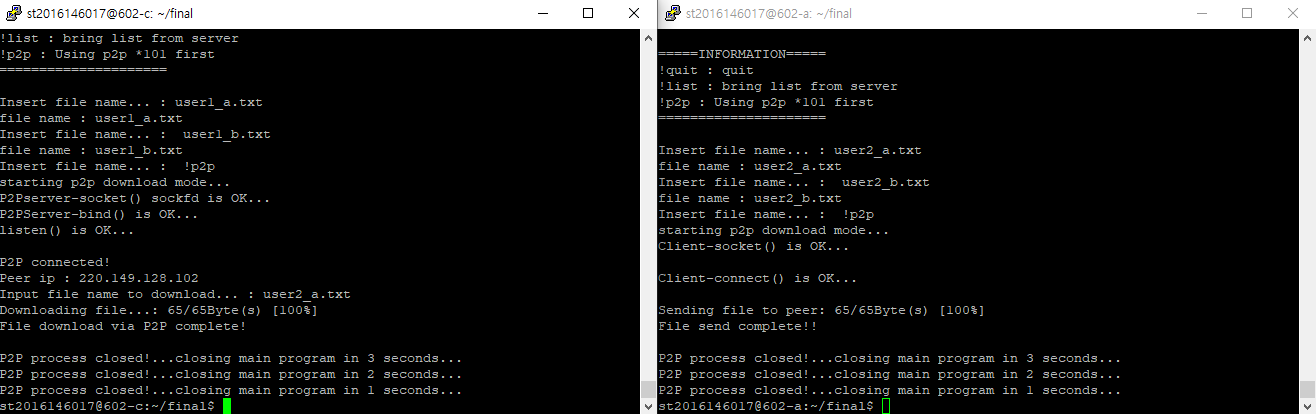
파일 리스트가 서버에 정상적으로 작동, 소유자의 정보가 출력, server\_file.lst에 저장됨.

(P2P 모드 진입)

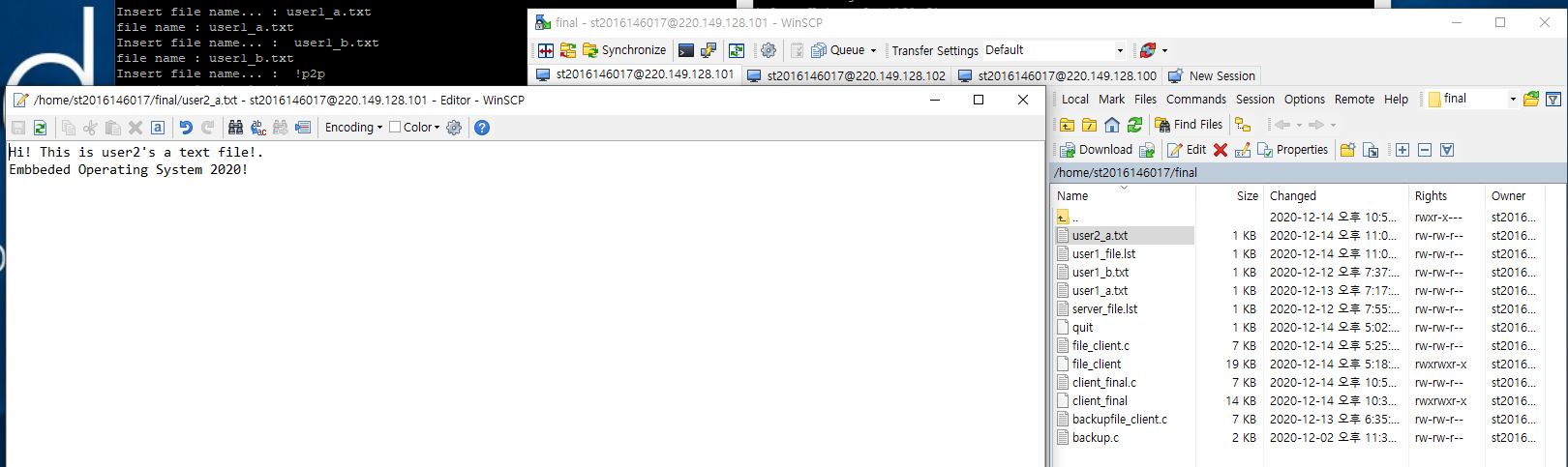


서버 역할을 하는 101을 먼저 p2p모드로 진입시키고 102를 뒤이어 진입시킴. 101의 창(좌측 하단)에 P2P가 연결되었다는 문구와 peer의 ip가 출력, 파일명을 입력하라는 문구가 나옴.

(올바른 파일명 입력한 경우)

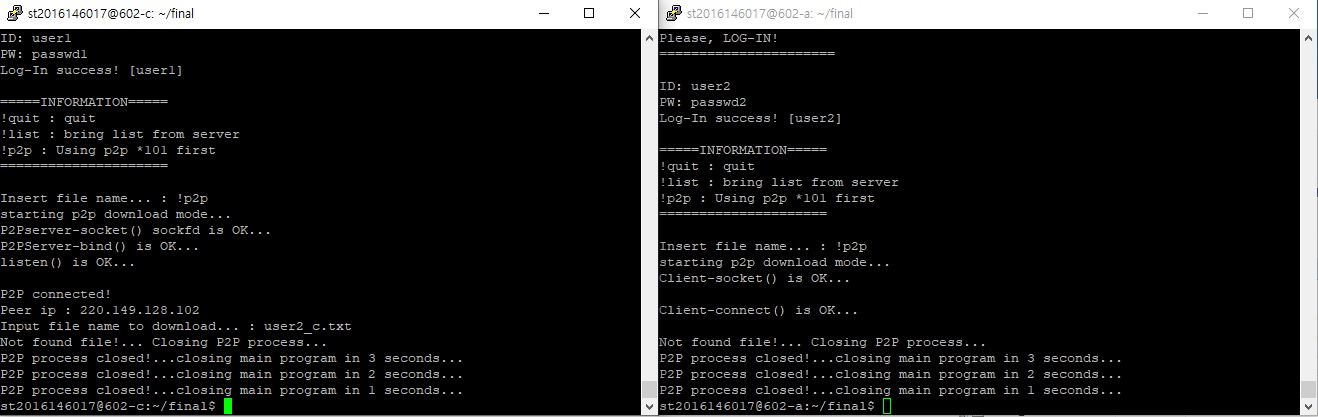


User2(102)에 존재하는 user2\_a.txt 파일을 입력한 결과. 102는 성공적으로 전송했다는 문구가 나오고 프로그램이 종료되었고 101은 성공적으로 다운로드 후 프로그램이 종료되었다.



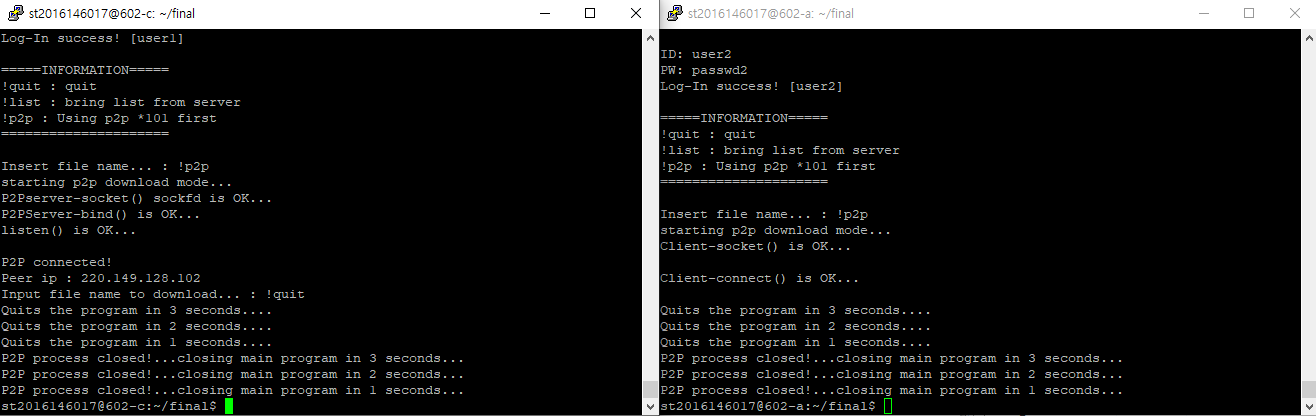
Winscp로 확인한 결과. 101의 파일 중 다운로드 받은 user2\_a.txt의 내용이 깨지지 않고 잘 다운로드 된 모습을 볼 수 있다.

(잘못된 파일명 입력한 경우)



User2(102)에 존재하지 않는 파일명인 user2\_c.txt 를 입력한 경우. 양쪽 모두에 nofile 이 출력되고 프로그램이 종료되었다.

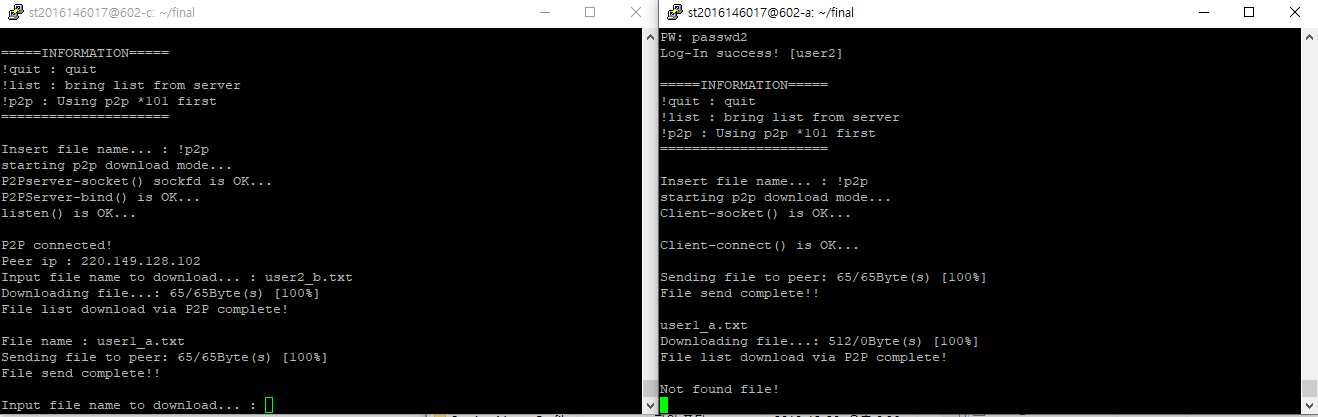
(!quit 명령어를 입력한 경우)



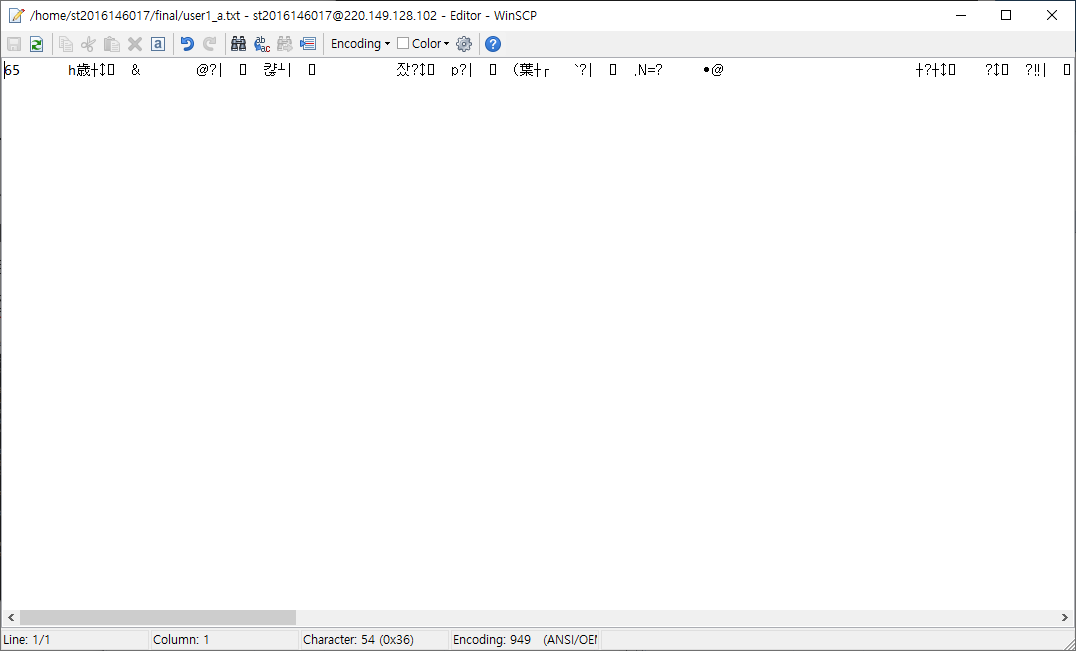
102로 !quit 명령어가 전달되어 101,102 모두 quit 동작, 자식 프로세스 종료 후 부모 프로세스가 종료되는 모습.

고찰사항

처음 프로젝트를 진행할 때 project#3의 내용은 102->101로 파일 전송을 하고 프로그램이 종료되는 것이었지만 저희 조는 더 응용, 발전시켜서 101이 파일을 받은 뒤 102가 파일명을 입력, 101에게 전송하면 101도 파일명을 비교, 전송하는 프로그램을 짜고 싶었으나 101이 파일명을 받고 정상적으로 102에게 보냈지만 102가 buf의 쓰레기값을 받아와 진행이 되지 않았습니다.



101이 102에게서 파일을 잘 받았고 101이 102에게 user1\_a.txt 파일 전송을 성공적으로 진행했으나 102가 잘못된 값을 받아와 파일이 손상되었음.



손상된 user1\_a.txt의 모습.

이를 수정하고 싶었으나 시간이 부족했고 미처 완성하지 못해 너무 아쉬웠습니다.

참고자료

파일 전송(sender, receiver)

<https://twinw.tistory.com/155>

wait 함수, <sys/wait.h> 헤더

<https://codetravel.tistory.com/30>