FTP Server Programming Assignment 1

2018068622 임문경

Server argument : control port number / data port number (optional)

ex) java FTPServer.java 20 21

default argument : 2020 / 2121

Client argument : server ip / control port number / data port number (optional)

ex) java FTPClient.java 192.168.0.2 20 21

default argument : 127.0.0.0 2020 2121

**Client – Server Commend Channel Connection ~ Commend read/write**

1. server는 자신의 control port의 welcome socket을 생성하여 client의 요청을 기다린다.

또한 server는 현재 directory를 나타내는 currentDirectory변수를 미리 선언.

2. client는 server의 control port로 연결을 요청, server는 해당 요청에 대해서 별도의 connection socket을 사용하여 연결을 완료하며 통신을 위해 각각의 input stream과 output stream line을 생성( inFromClient / outToClient / outToServer / inFromServer )

3. stream line을 생성한 후 clinet가 outToServer stream을 통해 명령어 전송

4. server는 inFromClient stream을 통해서 명령어를 받은 후 입력받은 내용을 명령어부분과 , 만약 존재한다면, 경로부분을 분리하여 각각 commend[0] , commend[1] 내부에 저장

경로부분(commend[1])을 route변수에 따로 저장해서 관리한다.

지속적인 명령어의 입력을 위해서 3,4번 단계를 while(true)문 내부에 작성한다.

**CD Commend**

1-1. “CD + 주소” case

1. 주소가 절대경로인 경우, route값을 currentDirectory변수에 그대로 대입
2. 주소가 절대경로가 아니거나 “CD .” , “CD ..” 명령어인 경우, currentDirectory값에 주소를 연결시킨다.
3. 주소가 유효하지 않은 경우 입력하기 전의 값으로 돌아가야 하므로 currentDirectory값을 변경하기 전 유효한 주소인지 검사후 currentDirectory값을 변경한다.

1-2. “CD” case

1. currentDirectory값을 그대로 File class의 argument로 생성한다.

2. server는 currentDirectory를 arugment로 생성된 File class의 유효성을 검사하여 결과에 따른 status code와 status message를 sendString 변수에 저장한다.

3. server console에 client로 부터 받은 요청 메세지와 자신이 보낼 응답 메세지(sendString)를 띄운 후 client에게 응답 메세지를 전송한다.

4. client는 server로부터 응답메세지를 읽은 후 status code가 200(success)라면 현재 경로를 출력, 그게 아니라면(501 : fail) 오류 메세지를 출력한다.

**QUIT Commend**

client가 QUIT명령어를 입력하면 server와 client 모두 열려있는 stream을 닫고 반복문을 빠져나간다.

**LIST Commend**

1-1. “LIST + 주소” case

1. 주소가 절대경로인 경우, route값을 File class argument로 File class 생성.
2. 주소가 절대경로가 아니거나 “LIST .” , “LIST ..” 명령어인 경우, currentDirectory값에 route값을 연결시킨후 File class생성.

1-2. “LIST” case

1. currentDirectory값을 그대로 File class의 argument로 생성한다.

2. server는 생성된 File class의 유효성을 검사하여 결과에 따른 status code와 status message를 sendString 변수에 저장 후 Client에게 전송.

3. 경로가 유효하다면 server는 해당 경로의 파일목록을 sendString 변수에 모두 연결시켜 client에게 전송한다.

4. Client는 Server로부터 응답메세지를 읽은 후 status code가 200이라면 server로부터 파일목록을 받아 출력한다. 그게 아니라면(501 : fail) 오류메세지를 출력한다.

5. server는 모든 전송을 끝내면 server console에 요청메세지와 응답메세지를 출력

**GET Commend**

1-1. “GET + 절대경로” case

route값을 그대로 File class argument로 File class 생성

1-2. “GET+ 상대경로“ case

currentDirectory값에 route값을 연결시켜 argument로 File class 생성

2. server는 생성된 File class의 유효성을 검사하여 결과에 따른 status code와 status message를 sendString 변수에 저장 후 Client에게 전송.

3-1. 경로가 유효한 경우(status code == 200)

1. server는 파일전송을 위한 data channel TCP socket을 생성하여 client로부터 연결을 기다린다.
2. client가 server로부터 응답메세지를 받은후 status code가 200이라면 client console에 파일이름과 파일크기를 출력한다.
3. client가 data channel TCP socket을 생성해 server에게 연결을 요청한다.
4. 연결이 된 후 server와 client는 파일입출력과 전송을 위한 stream을 생성

(server : input / inACK / output , client : input / outACK / output)

server – input : file을 입력받기 위한 FileInputStream

server – inACK : client로부터 ACK를 받기 위한 InputStream

server – output : client에게 file을 전송하기 위한 OutputStream

client – input : server로부터 file을 전송받기 위한 InputStream

client – outACK : server에게 ACK를 전송하기 위한 OutputStream

client – output : file을 생성 및 출력하기 위한 FileOutputStream

1. server는 전송을 위해 총 1004byte의 chunk를 생성하여 0~4번 인덱스에 각각 sequence number, checksum 그리고 data chunk크기를 저장한 후, 나머지 공간에 file내용을 입력받아 client에게 전송한 후 client로부터 ACK메세지를 기다린다.
2. client는 chunk에 저장된 sequence number를 1증가시킨 ack number와 checksum값을 ACK패킷에 저장한 뒤 server에게 ACK를 전송
3. chunk의 data부분을 저장후 console에 ‘#’를 출력한 후 다시 server로부터 입력을 기다린다.
4. server는 client로부터 ACK메세지를 받은후 chunk size와 sequence number를 업데이트하고 chunk를 다시 전송하며, 모든 파일내용을 전송할때까지 위의 과정을 반복한다.

이때, sequence number는 client로부터 받은 ACK number이다.

1. client는 server로부터 모든 파일내용을 전달받으면 console에 완료메세지를 출력한 한다. 그 후 server와 client는 모든 stream을 종료한다.

3-2. 경로가 유효하지 않은 경우(status code ≠ 200)

1. client는 console에 오류메세지를 출력한다

4. 모든 전송이 끝난 후 server는 요청메세지와 응답메세지를 console에 출력한다.

**PUT Commend**

1. server는 client로부터 입력받은 파일이름을 route로하여 currentDirectory에 연결하여 File class를 생성한다.

2. client는 현재 자신의 directory내에서 명령어로 입력한 파일이름을 argument로 File class를 생성한 후 server에게 파일크기를 전송한다.

이 때 File class가 존재하지 않거나 file이 아닌 경우 파일크기를 0으로하여 전송한다.

3-1. 정상적인 파일크기 및 유효한 argument를 받은 경우

1. server는 파일전송을 위한 data channel TCP socket을 생성하여 client로부터 연결을 기다린다.
2. client가 server로부터 응답메세지를 받은후 status cose가 200이라면 data channel TCP socket을 생성해 server에게 연결을 요청한다.
3. 연결이 된 후 server와 client는 파일입출력과 전송을 위한 stream을 생성한다.

(server : input / output / outACK , client : input / output / inACK)

server – input : client로부터 file을 전송받기 위한 InputStream

server – output : server directory에 file을 저장하기 위한 FileOutputStream

server – outACK : client에게 ACK를 전송하기 위한 OutputStream

client – input : file을 입력받기 위한 FileInputStream

client – output : server에게 file을 전송하기 위한 OutputStream

client – inACK : server로부터 ACK를 받기위한 InputStream

1. client는 전송을 위해 총 1004byte의 chunk를 생성하여 0~4번 인덱스에 각각 sequence number, checksum 그리고 data chunk크기를 저장한 후, 나머지 공간에 file내용을 입력받아 server에게 전송한 후 console에 ‘#’를 출력하고 server로부터 ACK메세지를 기다린다.
2. server는 client로부터 chunk를 수신받은 후, chunk에 저장된 sequence number를 1증가시켜 ack number을 업데이트하고 checksum값을 ACK패킷에 저장한 뒤 client에게 ACK를 전송
3. 그 후 server는 chunk의 data부분을 currentDirectory에 저장한다.
4. client는 server로부터 ACK메세지를 받은후 ACK메세지의 ACK number를 sequence number로하여 chunk를 업데이트하고 data chunk도 다시 업데이트한 후 server에게 chunk를 전송하며, 모든 파일을 전송할때까지 위의 과정을 반복한다.
5. client는 server로부터 모든 파일내용을 전달한 후 console에 완료메세지를 출력한 한다. 그 후 server와 client는 모든 stream을 종료한다.

3-2 client로부터 파일크기를 받은 server는 파일크기가 0인경우나 파일의 이름이 입력되지 않은 경우

1. server는 client에게 status code가 501인 메세지를 전달한다.
2. 오류메세지(status code == 501)를 받은 client는 console에 오류메세지를 출력한다.

4. 모든 전송이 끝난 후 server는 요청메세지, 파일크기 그리고 응답메세지를 console에 출력한다.

**data chunk size 저장방식**

GET / PUT Commend에서 data chunk size는 chunk[3],chunk[4]내에 2진법형태로 저장되어 있으므로 해당 값을 읽기 위해서 byte -> int -> 2진수 string -> int로 변환하여 값을 읽어올 수 있었다.

Integer.*parseInt*(Integer.*toBinaryString*(Byte.*toUnsignedInt*(chunk[3]))+Integer.*toBinaryString*(Byte.*toUnsignedInt*(chunk[4])),2);

변환하는 과정을 코드로 작성했을 경우 위와 같은 명령어를 통해서 최종적으로 Int형태의

**programme 실행 방식**

1. 실행환경 windows 10

windows PowerShell

java version "15.0.1”

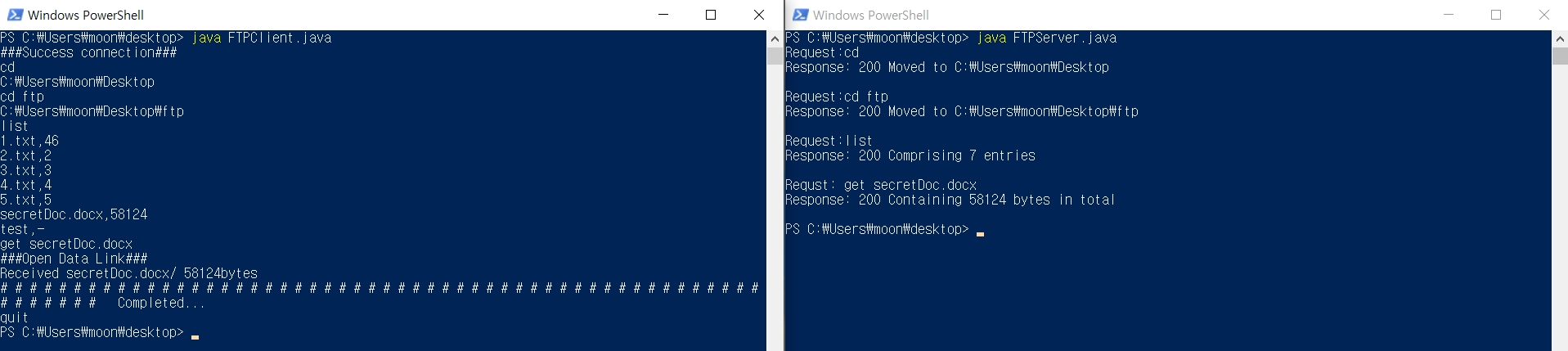
Java(TM) SE Runtime Environment (build 15.0.1+9-18)

2. 실행방법

1. “java FTPServer.java”명령어로 server를 구동시킨다. 만약 필요하다면 인자를 입력해 port번호를 지정할 수 있다.
2. “java FTPClient.java”명령어로 client를 구동시켜 FTP server에 연결시킨다. 만약 필요하다면 인자를 입력해 server IP와 port번호를 지정할 수 있다.

3. 실행예시

**[server]**

****

**[client]**

