네트워크 – FTP Client and Server Programing

- assignment 1 -

소프트웨어전공 2013010926 강은석

1. 과제 설명

: FTP 프로토콜 서버 및 클라이언트 프로그램을 Java blocking IO를 사용하여 구현한다. 한 클라이언트가 서버와 연결되어 수행 중인 경우 다른 클라이언트는 연결 요청에서 블록되고 대기하다가 선 클라이언트가 수행을 종료하면 바로 ftp 세션이 시작될 수 있도록 한다.

: 클라이언트가 파일 송수신을 위해서 "LIST", "GET", "PUT", "CD"와 같은 명령어를 요청할 수 있다.

- LIST: 파일 서버의 특정 디렉토리의 파일 보는 명령어
- GET: 파일 서버로부터 파일을 download 하는 명령어
- PUT: 파일 서버에 파일을 upload 하는 명령어
- CD: 파일 서버에서 현 디렉토리 위치를 변경하는 명령어

: 기본적으로 절대 경로, 상대 경로, "."(현재 위치), ".." (상위 디렉토리)를 지원해야 한다.

2. 프로그램 설계와 구조

: Server는 항상 Welcome Socket으로 Client의 소켓과의 연결을 기다립니다. Client에서 지정한 Ip와 Port Number로 Socket을 생성하고 서버의 Socket과 accept가 되면 Connection이 생기고 Client로부터 "CD, LIST, GET, PUT"과 같은 파일 송수신을 위한 명령어로 request가 들어오면 Server는 각 명령어에 맡는 처리 작업을 진행한 뒤 response를 Client에게 전달해 줍니다. 이 후 Client로부터 "Done"이라는 명령어가 Server로 수신이 되면 Server와 Client간 connection을 끊고 다른 Client와의 연결을 기다리도록 설계하였습니다.

3. 동작 절차 및 구현 방법

```
BufferedWriter bufWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
sentence = inFromUser.readLine();
if ("".equals(sentence)) {
  System.out.println(errorWrongcmd);
  continue;
}
sentenceWords = sentence.split("\\s");
bufWriter.write(sentence);
bufWriter.newLine();
bufWriter.flush();
```

그림 1 client에서 명령어 전달

```
BufferedReader in = new BufferedReader( new InputStreamReader( clientSocket.getInputStream()));
request = in.readLine();
String[] requestWords = request.split("\\s");
// for disconnect to client and server
```

그림 2 server에서 명령어 수신

기본적인 CD, LIST, GET, PUT 명령어는 Client에서 BufferedWriter.write()로 Server로 전달해주고 Server에서는 BufferedReader.readLine() 으로 수신합니다.

1) CD/LIST

```
if ("CD".equals(sentenceWords[0])) {
    InputStream in = socket.getInputStream();
    DataInputStream din = new DataInputStream(in);
    int statusCode = din.readInt();
    int responseLength = din.readInt();
    String response = din.readUTF();
    System.out.println(response);
  }
```

그림 3 client에서 CD 명령어 response 수신

```
else if("CD".equals(requestWords[0])) {
  OutputStream out = clientSocket.getOutputStream();
  DataOutputStream dout = new DataOutputStream(out);
  int statusCode = 1;
  // response about "CD" or "CD ." request
  if(requestWords.length == 1 || ".".equals(requestWords[1])) {
    //String currentPath = new java.io.File("").getCanonicalPath();
    String currentPath = System.getProperty("user.dir");
    dout.writeInt(statusCode);
    dout.writeInt(currentPath.length());
    dout.writeUTF(currentPath);
  }
```

그림 4 Server에서 CD명령어에 대한 response

CD와 LIST인 경우가 비슷해서 묶어서 설명하겠습니다. 우선 그림4을 보면, Server에서 CD/LIST 요청에 따라 DataOutputStream을 이용해서 statusCode, responseLength, response를 Client에 보내주고 그림 3에서 보면 Client는 DataInputStream를 통해서 각 response를 수신합니다.

2) GET/PUT

```
// request about "GET" and get response
else if("GET".equals(sentenceWords[0])) {
  int len, data,statusCode;
  byte[] buffer = new byte[1024];
  InputStream in = socket.getInputStream();
  DataInputStream din = new DataInputStream(in);
  statusCode = din.readInt();
  data = din.readInt();              // get file size
  String filename = din.readUTF();  // get file name
  if (statusCode == -1) {           // error response
    System.out.println(filename);
    continue;
  }
  FileOutputStream out = new FileOutputStream(filename);
  for(int i = data; i > 0; i--){    // save file to Client
    len = in.read(buffer);
    out.write(buffer,0,len);
  }
```

그림 5 client에서 GET 명령어 response 수신

```
else if("GET".equals(requestWords[0])) {
  OutputStream out = clientSocket.getOutputStream();
  DataOutputStream dout = new DataOutputStream(out);
  int statusCode = 1;
  String currentPath = System.getProperty("user.dir");
```

```
byte[] buffer = new byte[1024];
int len;
int data=0;
File file = new File(currentPath + "\\" +requestWords[1]);
if (file.isFile()) {
  FileInputStream fin = new FileInputStream(currentPath + "\\" +requestWords[1]);
  String fileName = requestWords[1];
  while((len = fin.read(buffer))>0){  // calculate file size
      data++;
    }
  fin.close();
  fin = new FileInputStream(currentPath + "\\" +requestWords[1]);
  dout.writeInt(statusCode);
  dout.writeInt(data);
  dout.writeUTF(fileName);
  len = 0;
  // send file to client
  for(int i = data; i > 0; i--){
    len = fin.read(buffer);
    out.write(buffer,0,len);
  }
  fin.close();
  out.flush();
```

그림 6 Server에서 GET명령어에 대한 response

GET와 PUT인 경우가 비슷해서 묶어서 설명하겠습니다. 우선 그림6을 보면, Server에서 GET요청에 따라 DataOutputStream과 FileInputStream을 이용해서 statusCode, fileSize, fileData를 Client에 보내주고 그림 5에서 보면 Client는 DataInputStream과 FileOutputStream를 통해서 각 response를 수신하고 파일을 저장합니다. 그림 6에 Server에서 buffer크기를 할당해서 파일을 읽어서 file의 크기를 파악하고 file을 Client로 보내는데 buffersize를 1024로 정한 이유를 os에서 파일 크기를 보여줄 때 kbyte 기준으로 보여주기 때문에 1024로 정했습니다.

4. 컴파일 방법, 실행방법

그림 7 Compile FtpServer.java

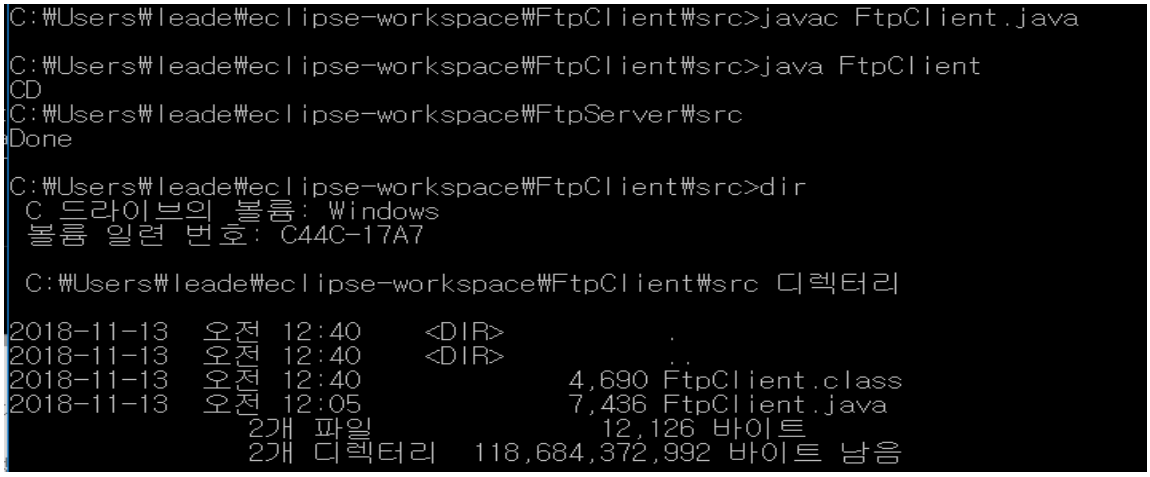

그림 8 Compile FtpClient

위의 그림과 같이 "FtpServer/src" 경로에 들어가서 명령어 "javac FtpServer.java"를 입력하고 "FtpClient/src" 경로에서 명령어 "javac FtpClient.java"를 입력하면 각 FtpServer.class, FtpClient.class 가 생기는 것을 확인 할 수 있습니다.

서버와 클라이언트 경로에서 java <class 이름> <port number>를 입력하면 실행이 됩니다.

이후 각 CD , LIST, GET, PUT에 대한 설명을 이어가겠습니다.

1) CD

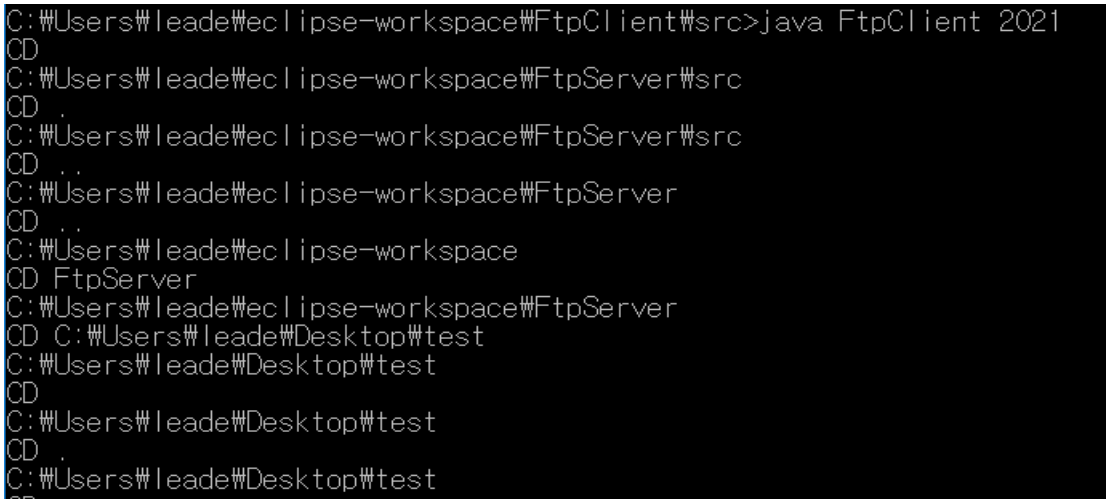

그림 9 CD 명령어 실행

CD 명령어 예시를 보면 "CD", "CD ."에 대해서 현재 working directory path를 보여줍니다.

"CD .."은 상대경로를 지원하며 현재 working directory의 부모 path로 이동합니다.

"CD Path"는 절대 경로를 지원하며 Path에 해당하는 path로 이동합니다.

2) LIST

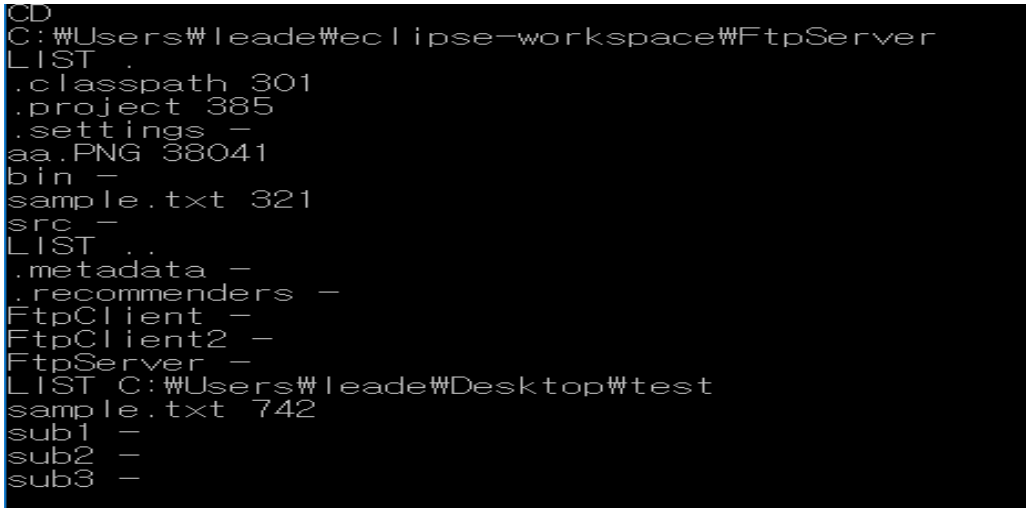

그림 10 LIST 명령어 실행

LIST 명령어 예시를 보면 "LIST ."에 대해서 현재 working directory path의 파일 목록을 보여줍니다.

"LIST .."은 상대경로를 지원하며 working directory의 부모의 파일 목록을 보여줍니다.

"LIST Path"는 절대 경로를 지원하며 Path에 해당하는 path로 이동합니다.

위의 그림에서 파일은 옆에 파일 크기가 나오며, directory인 경우는 "-"를 보여줍니다.

3) GET

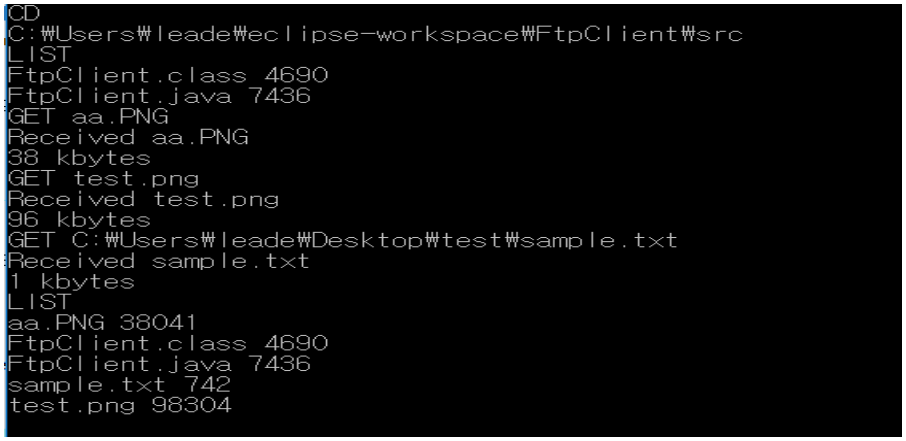

그림 11 GET 실행

그림 5에서 "FtpServer₩src" 폴더에서 상대 경로로 GET "aa,PNG", "test.PNG"를 할 수 있고 절대 경로로 "GET C:₩Users₩leade₩Desktop₩test₩sample.txt"를 하고 "FtpClient₩src" 폴더를 LIST로 확인하면 GET으로 받아온 파일들이 있는 것을 확인 할 수 있습니다. 위에서 보시는 것과 같이 GET을 상대 경로로 받아올 수 있는 폴더는 서버가 있는 "FtpServer₩src"폴더가 기준입니다. 그리고 Client에서 GET으로 받아온 파일들은 "FtpClient₩src"에 저장된다는 것을 확인 할 수 있습니다.

4) PUT

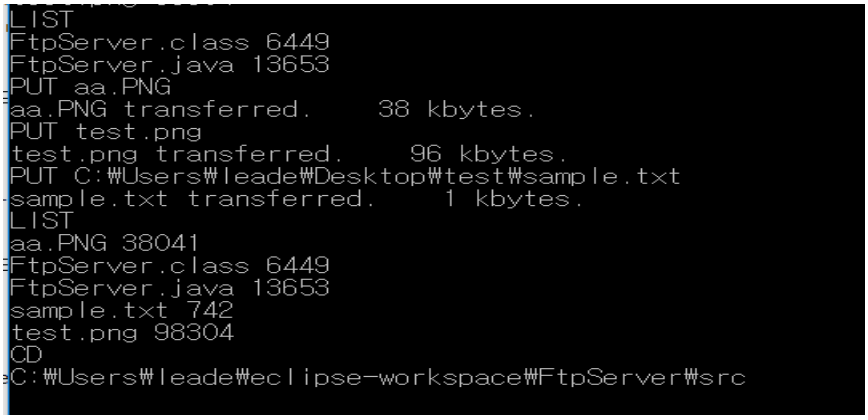

그림 12 PUT 실행

그림 6에서 "FtpClient₩src" 폴더에서 상대 경로로 PUT "aa,PNG", "test.PNG"를 할 수 있고 절대 경로로 "PUT C:₩Users₩leade₩Desktop₩test₩sample.txt"를 하고 "FtpServer₩src" 폴더를 LIST로 확인하면 GET으로 받아온 파일들이 있는 것을 확인 할 수 있습니다. 위에서 보시는 것과 같이 PUT으로 파일을 서버로 upload 하는 폴더는 클라이언트가 있는"FtpClient₩src" 폴더가 기준입니다. 그리고 Client에서 PUT으로 upload한 파일들은 "FtpServer₩src"에 저장된다는 것을 확인 할 수 있습니다.