Computer Network 2017

FTP Client & Server Implementation

2012017850 박 창 대

1. 개발 환경

프로그램 구현은 다음과 같은 환경에서 진행되었으며 사용 OS 는 리눅스를 기준으로 하였다.

Java 버전

```
changdae@changdaeui-MacBook-Pro:~/Desktop/source/Java/FTP_Server&Client/bin$ java -version
java version "1.8.0_112"
Java(TM) SE Runtime Environment (build 1.8.0_112-b16)
Java HotSpot(TM) 64-Bit Server VM (build 25.112-b16, mixed mode)
```

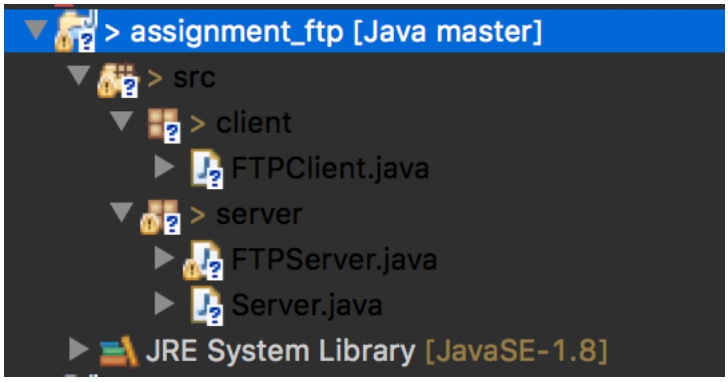
Eclipse 버전

```
Eclipse IDE for Java Developers

Version: Neon.2 Release (4.6.2)
```

2. 컴파일 및 실행

- 먼저 이클립스에서 Java 프로젝트를 생성한다.
- 압축 해제된 ftp_src 폴더 안에 있는 client, server 폴더를 방금 만든 프로젝트의 src 폴더 안에 복사한다.



- FTPServer.java 파일을 더블 클릭하여 창을 띄운 후, Run 버튼을 클릭하여 서버를 실행한다.
- 터미널을 열고 해당 프로젝트의 bin 폴더로 이동한다

```
changdae@changdaeui-MacBook-Pro:~/Desktop/source/Java/assignment_ftp/bin$ ls
client server
```

- java client.FTPClient 를 통해 클라이언트를 실행한다.

```
changdae@changdaeui-MacBook-Pro:~/Desktop/source/Java/assignment_ftp/bin$ java client.FTPClient
**********************************************************************************************************************************

CD [path] => Change current working directory to [path]. Both relative and absolute path & '.', '..' are available.
LIST [path] => List all files and directories in [path]. Both relative and absolute path & '.', '..' are available.
GET [path] => download file from [path]. Both relative and absolute path are available.
PUT [file] => upload file located in client's current working directory to server's current working directory.
q => Exit.

**********************************************************************************************************************************
```

- 이후, port 번호를 직접 지정하고 싶다면 기존의 서버 프로세스를 종료한 뒤에 새로운 터미널을 열어 다음과 같이 서버를 실행한다.

```
changdae@changdaeui-MacBook-Pro:~/Desktop/source/Java/assignment_ftp/bin$ java server.FTPServer 8887 8888
```

- 그리고 클라이언트를 다음과 같이 실행한다.

```
changdae@changdaeui-MacBook-Pro:~/Desktop/source/Java/assignment_ftp/bin$ java client.FTPClient 8887 8888
********************************************************************************************************************************

CD [path] => Change current working directory to [path]. Both relative and absolute path & '.', '..' are available.
LIST [path] => List all files and directories in [path]. Both relative and absolute path & '.', '..' are available.
GET [path] => download file from [path]. Both relative and absolute path are available.
PUT [file] => upload file located in client's current working directory to server's current working directory.
q => Exit.

********************************************************************************************************************************
```

3. 프로그램 작동 시현

- CD

```
CD /Users/changdae/Desktop
OK - /Users/changdae/Desktop/

CD source
OK - /Users/changdae/Desktop/source/

CD ..
OK - /Users/changdae/Desktop/

CD .
OK - /Users/changdae/Desktop/

CD nodir
FAILED - Directory name is invalid.
```

- LIST

```
LIST assignment_ftp
.classpath,295
.gitignore,6
.project,373
.settings,-
bin,-
src,-
```

```
LIST.
.classpath,226
.DS_Store,6148
.project,371
.git,-
.metadata,-
.recommenders,-
03_10,-
assign_ftp,-
assignment_ftp,-
FTP_Server&Client,-
JSPTest,-
Minsuk_Changdae,-
programming_assignment_1,-
Programming_Assignment_3_Pizza_Shop,-
Programmiong_Assignment_2_Election_Simultation,-
quiz_2,-
RemoteSystemsTempFiles,-
Servers,-
SocketPractice,-

LIST ..
.DS_Store,14340
2017_ITE2038_2012017850,-
CLionProjects,-
Java,-
PycharmProjects,-
r_project,-
rails_project,-
ruby_project,-
sublime,-
```

```
LIST /Users/changdae/Desktop/temp
Empty Directory.

LIST nodir
Directory name is invalid.
```

- GET

```
GET /Users/changdae/Desktop/hello.txt
Receiving data. Please wait..
Received hello.txt / 25byte(s)

GET /Users/changdae/Desktop/pic.png
Receiving data. Please wait..
Received pic.png / 34010byte(s)
```

```
CD .
/Users/changdae/

GET Desktop/hello.txt
File already exists. (Client)

GET nofile
File does not exist. (Server)
```

```
changdae@changdaeui-MacBook-Pro:~/Desktop/source/Java/assignment_ftp/bin$ ls
client    hello.txt pic.png   server
```

- PUT

```
CD .
/Users/changdae/Desktop/temp/

LIST .
Empty Directory.

PUT hello.txt
Sent hello.txt / 25byte(s)

PUT pic.png
Sent pic.png / 34010byte(s)

LIST .
hello.txt,25
pic.png,34010

PUT pic.png
File already exists. (Server)

PUT nofile
File does not exist. (Client)
```

```
changdae@changdaeui-MacBook-Pro:~/Desktop/temp$ ls
hello.txt pic.png
```

4. 프로그램 구조 및 동작 절차

프로그램은 FTPClient, FTPServer, Server, 이렇게 총 3 개의 class 로 구성되어 있다. FTPClient 는 command 와 data 에 대한 소켓과 사용자의 입력에 대한 스트림, 서버에 대한 data 및 command 입출력 스트림을 가지고 있다. FTPServer 에는 Data 와 Command 전용 소켓을 위한 웰커밍 소켓을 대기시킨다. 이후 FTPClient 가 실행되면, FTPServer 의 웰커밍 소켓은 각각 .accept() 되며 accept 메소드의 리턴 값인 Socket 객체를 매개변수로 하여 Server 객체를 생성한다. Server 객체의 생성자는 전달받은 Socket 객체를 통해 command 와 data 에 대한 connection socket, 입출력 스트림을 초기화한다. 그리고 start 메소드를 통해 쓰레드를 실행한다. 생성된 쓰레드는 가장 먼저 클라이언트로부터 command 가 입력되기를 기다린다. 클라이언트의 입력에 따라 적절한 메소드를 실행하며 각 메소드의 구체적인 동작 절차는 다음과 같다.

- 클라이언트가 CD 명령을 입력하면 서버 사이드에서는 전달받은 클라이언트의 입력을 매개변수로 하여 changeDirectory 메소드를 실행한다. 이 메소드는 클라이언트 입력이 절대경로인 경우, ‘CD ..’ 인 경우, ‘CD .’ 인 경우, 상대경로인 경우에 따라 다르게 실행된다. 절대경로로 입력된 경우에는 클라이언트의 커맨드에서 ‘CD’를 제거하여 디렉토리 경로를 파싱한 후 해당 디렉토리가 존재하는지 확인한다. 만약 존재한다면 System.setProperty 메소드를 통해 서버 프로세스의 working directory 를 파싱된 디렉토리로 변경한다. ‘CD ..’ 가 입력된 경우에는 setProperty 메소드의 매개변수를 상위 디렉토리의 경로를 지칭하는 ‘new File(‘..’).getCanonicalPath()’로 하여 working directory 를 변경한다. ‘CD .’인 경우에는 아무것도 하지 않는다. 상대경로인 경우에는 상대경로를 클라이언트의 입력으로부터 파싱하여 완전한 절대경로를 만들어 setProperty 를 실행한다. 이처럼 각각의 경우에 따라 적절한 수행이 이루어지고 나면 현재 working directory 의 경로를 반환한다.
- LIST 명령이 입력되면 listFilesDirectories 메소드가 실행된다. 이 메소드도 위와 마찬가지의 방법으로 먼저 ‘LIST ’, ‘LIST ..’, 상대경로인 경우에 경로를 완전한 절대경로로 만들어준다. 그 다음, 이를 통해 해당 directory 존재 유무를 검사한다. 존재한다면 해당 경로를 매개변수로 하여 File 객체를 선언해주고 listFiles 메소드를 통해 반환되는 파일 객체들을 File 객체 배열에 저장한다. 그리고 StringBuilder 객체를 2 개 선언하고, for 문을 통해 배열에 저장된 File 객체가 파일인 경우와 디렉토리인 경우로 나누어 각기 다른 StringBuilder 객체에 파일 및 디렉토리의 이름을 append 한다. 반복문이 끝나면 두 StringBuilder 객체를 append 하고 완성된 문자열을 반환한다.

CD 또는 LIST 명령에 대한 메소드가 종료되면 서버는 해당 메소드가 반환한 문자열에 ‘[EndOfData]₩n’를 더해 클라이언트에게 전달한다. 이와 같이 해주는 이유는 클라이언트가 LIST 명령과 같이 여러 줄의 문자열이 반환될

수 있는 경우를 대비하여 while 문으로 계속해서 서버로부터 문자열을 readLine 하고 있기 때문에, 전송되는 문자열의 끝을 알려주어 적절하게 반복문을 종료할 수 있도록 하기 위함이다.

- GET 명령이 입력되면 클라이언트에서는 receiveData, 서버에서는 snedData 메소드가 실행된다. 먼저 클라이언트는 현재 working directory 에 받고자 하는 파일이 이미 존재하는지 검사한 뒤, 존재하지 않는다면 서버로부터 파일 사이즈 응답을 기다린다. 서버에서는 파일 존재 유무를 검사하는데, 만약 파일이 존재하지 않는다면 파일의 크기 대신에 'NOT EXISTS₩n'라는 문자열을 클라이언트에 전송한다. 이 경우에 클라이언트는 콘솔에 파일이 존재하지 않는다는 문구를 띄우고 메소드를 종료한다. 파일이 존재하면 서버는 클라이언트에게 파일사이즈를 전송하고 파일 스트림을 통해 파일의 내용을 byte 배열에 저장한다. 그리고 byte 배열 내용 전부를 클라이언트에게 전송한다. 클라이언트는 전송받은 파일 크기 만큼 byte 배열을 준비하고 서버로부터 전송받은 데이터를 해당 배열에 저장한다. 그리고 그 내용을 파일에 작성한다.
- PUT 명령이 입력되면 이번에는 반대로 클라이언트에서는 sendData, 서버에서는 receiveData 메소드가 실행된다. 먼저 클라이언트는 현재 보내고자 하는 파일이 현재 working directory 에 정상적으로 존재하는지 검사한다. 존재한다면 커맨드 채널을 통해 서버에게 입력된 커맨드를 전송한다. 그리고 클라이언트는 서버에 해당 파일이 이미 존재하는지 여부에 대한 응답을 대기한다. 서버는 해당 파일이 현재 working directory 에 존재하는지 검사하고, 만약 이미 있다면 클라이언트에게 'ALREADY EXISTS₩n' 라는 응답을 보낸다. 이 경우 클라이언트는 적절한 문구를 콘솔에 출력하고 메소드를 종료한다. 전송하고자 하는 파일이 서버의 현재 working directory 에 없다면 서버는 클라이언트에게 OK 를 전송하고 클라이언트로부터 파일 사이즈를 응답 받을 때까지 대기한다. 클라이언트는 전송할 파일의 사이즈를 서버에게 전송하고 byte 배열에 파일의 내용을 저장한뒤 서버에게 전송한다. 서버는 전송받은 파일 사이즈만큼 byte 배열을 선언하고 그 안에 클라이언트로부터 전달받은 데이터를 저장한다. 그리고 그 내용을 파일에 작성한다.

5. 과제를 통해 배운 점 및 구현상 미숙한 점

- TCP 기반의 소켓 프로그래밍의 큰 구조를 익힐 수 있었다.
- 쓰레드 활용의 의미를 이해할 수 있었고 Java 를 통해 기본적인 멀티쓰레드 프로그램을 작성하는 익힐 수 있었다.
- 2 주 정도 프로그램을 구현하면서 날마다 주석을 꼼꼼히 달아 두었던 것이 지난 시간동안 구현했던 내용을 금방 떠오르게 해주는 데에 많은 도움이 됐던 것 같다.
- startsWith, split, trim 등 Java 를 이용하여 파싱을 할 때 유용한 메소

드를 익힐 수 있었다.

- '+' 를 통해 문자열을 지속적으로 append 하는 것은 메모리와 퍼포먼스 측면에서 매우 비효율적이므로 StringBuilder 또는 StringBuffer 클래스를 사용해야 한다는 것을 알게 됐다.
- 프로그래밍을 할 때, 먼저 큰 그림에서 프로그램을 디자인하는 것을 마치고 구현을 진행해야 하는데, 디자인이 명확하지 않은 상태에서 구현을 진행하다 보니 이후에 구조를 바꿔야 하는 일이 발생하곤 했다. 구현 이전 단계의 디자인의 중요성을 다시 한 번 생각하게 됐다.
- 아직까지 Java 프로그래밍을 할 때 적절하게 클래스를 설계하는 것이 미숙한 것 같다는 생각이 들었다.
- 현재 프로그램 내부에서 서버의 웰커밍 소켓을 적절하게 close 하지 못하고 있다. 바람직한 프로그램의 소스 코드라면 소켓이나 스트림을 적절하게 close 해주어야 할 것인데 서버의 웰커밍 소켓만큼은 현재 프로그램 구조에서 어떻게 close 해주어야 할 지 모르겠다.
- 현재 구현된 바로는, 클라이언트가 'q'를 입력하면 클라이언트 프로세스는 종료된다. 이 때, 클라이언트의 커맨드를 기다리던 서버에서 발생하는 Exception 을 catch 해서 쓰레드를 종료하고 있는데, 이러한 방법이 바람직한 것인지는 확신이 서지 않는다.
- System.setProperty()를 통해 서버 프로세스의 working directroy 를 변경하고 있는데, 이렇게 하니 복수의 클라이언트가 서버의 working directory 변경을 실시간으로 공유하게 되는 문제가 생긴다. 즉, 클라이언트 1 이 CD 를 통해 디렉토리를 변경하면 클라이언트 2 에서도 서버의 working directory 가 변경되는 상황이 나타나는 것이다. 지금 생각해보니 System.setProperty()를 통해 프로세스의 working directory 를 실제로 변경하지 않고 클라이언트마다 자신이 보고 있는 서버의 working directory 의 경로를 저장하는 변수를 두어 서버와 데이터를 주고 받을 때 해당 변수도 항상 주고 받아서 작업을 처리했어야 진정한 멀티 쓰레드 FTP 프로그램이 될 수 있었을 것 같다는 아쉬움이 든다.