컴퓨터 네트워크 - Programming Assignment1

2019019016 서시언

<코드 설명 & 클라이언트-서버의 동작 절차>

1. Server와 Client 사이의 Command Channel

- Server
    1. command line argument로 주어진 port number(주어지지 않은 경우 2020)를 이용해 command channel을 위한 ServerSocekt을 만든다.
    2. while문 안에서 accept()를 이용해 클라이언트의 접속을 기다린다.
    3. accept()에서 리턴된 소켓에 대한 input stream인 inFromClient와 output stream인 outToClient를 만든다.
    4. 조건문이 exit==0인 while문을 하나 더 만들어서 inFromClient.readLine()을 이용해 클라이언트가 보낸 명령어를 받아 command에 저장한다.
- Client
    1. command line argument로 주어진 server host(주어지지 않은 경우 127.0.0.1)과 port number(주어지지 않은 경우 2020)를 이용해 서버와 연결된 Socekt을 만든다.
    2. 위의 소켓에 대한 input stream인 inFromServer와 output stream인 outToServer를 만든다.
    3. 조건문이 exit==0인 while문 안에서 유저의 입력을 받아 command에 저장하고, outToServer.writeBytes를 이용해 서버로 해당 command를 전송한다.
- 서버의 2번 과정 이후에 클라이언트의 1번 과정이 이루어지면, 서버의 3번 과정이 이루어진다.
- 클라이언트의 3번 과정 이후에 서버의 4번이 수행된다.
- 클라이언트로부터 받은 command를 " "를 기준으로 split했을 때 0번 값에 따라 아래의 LIST, GET, PUT, CD, QUIT가 수행된다.

2. LIST

-Server
    1. 클라이언트로부터 받은 command를 " "를 기준으로 split했을 때의 1번 인덱스의 값을 path에 저장한다.
        - 경로가 없는 경우 (LIST만 입력한 경우) 현재 디렉토리의 주소를 path에 저장한다.
        - 상대경로인 경우 changeToAbsolute 메소드를 이용해 경로를 절대경로로 바꿔준다.

    2-1. 해당 경로가 디렉토리가 아닌 경우 status code 501을 포함한 response를 보내고 break한다.

    2-2. 디렉토리인 경우, listFiles() 메소드를 이용해 해당 디렉토리의 파일 목록을 fileList에 저장한다.
    - 그 후 entry의 개수를 포함한, status code가 200인 response를 보내고, 각 entry의 이름과 크기 (디렉토리인 경우 '-')를 담은 response를 보낸다.

-Client
    1. inFromServer.readLine()을 통해 서버로부터 response를 받아온 뒤, 이를 " "를 기준으로 split해서 tmp에 저장한다.

    2-1. tmp[0]이 "200"이면 서버로부터 파일,디렉토리의 이름과 크기를 담고 있는 response를 받아와서 이를 출력한다.

    2-2. tmp[0]이 "501"이면 디렉토리의 이름이 잘못되었다는 메세지를 출력한다.
- 서버의 2번 이후에 클라이언트의 1번이 수행된다.

3. GET

-Server

1. 클라이언트로부터 받은 command를 " "를 기준으로 split했을 때의 1번 인덱스의 값을 path에 저장한다.
    - 경로가 없는 경우 (GET만 입력한 경우) status code 500을 포함한 response를 보내고 break한다.
    - 상대경로인 경우 changeToAbsolute 메소드를 이용해 경로를 절대경로로 바꿔준다.

2-1. 경로가 file이 아니라면 status code 401을 포함한 response를 보내고 break한다.

2-2. file이라면 file의 크기를 포함한, status code가 200인 response를 보낸 뒤, data channel을 위한 ServerSocket의 accept 메소드를 통해 클라이언트의 접속을 기다린다.
    - command line argument로 주어진 port number(주어지지 않은 경우 2121)을 이용해 data channel을 위한 ServerSocekt을 만든다.

1) accept()에서 리턴된 소켓에 대한 output stream인 dataToClient와 file에 대한 input stream인 dataFromFile을 만든다.

2) 3byte 크기의 byte array, header와 1000byte 크기의 byte array, data, 1003byte크기의 byte array, send_data를 만든다.

3) dataFromFile.read(data)를 통해 파일에 있는 테이터를 1000byte씩 읽어온다.

4) System.arraycopy 메소드를 이용해 header와 data의 내용을 send_data로 합쳐준다.

5) dataToClient.write를 이용해 send_data를 클라이언트로 전송해준다.

6) 파일을 모두 읽어서 클라이언트로 보낼 때까지 3)~5) 과정을 반복한다.

-Client

1. inFromServer.readLine()을 통해 서버로부터 response를 받아온 뒤, 이를 " "를 기준으로 split해서 tmp에 저장한다.

2-1. tmp[0]이 "200"이면 tmp[2](file의 크기)를 int형으로 바꿔 filesize에 저장한다.

1) command line argument로 주어진 server host(주어지지 않은 경우 127.0.0.1)과 port number (주어지지 않은 경우 2121)을 이용해 서버와 연결된 Socekt을 만든다.

2) 위의 소켓에 대한 input stream인 dataFromServer을 만든다.

3) command로 보냈던 파일의 경로명을 이용해 현재 디렉토리에 받아올 파일을 추가한 경로를 file에 저장한다.

4) 위의 file에 대한 output stream인 dataToFile을 만든다.

5) 3byte 크기의 byte array, header와 1000byte 크기의 byte array, data, 1003byte크기의 byte array, received_data를 만든다.

6) dataFromServer.read(received_data)를 이용해 서버에서 보낸 데이터 메시지를 받아온다.

7) System.arraycopy 메소드를 이용해 received_data의 내용을 header와 data로 나눠서 저장한다.

8) dataToFile.write 메소드를 이용해 data에 있는 내용을 파일에 써준다.

9) cntSize에 data의 크기를 더해준다.

10) cntSize가 fileSize와 같아질 때까지 6)~9)과정을 반복해준다.

2-2. tmp[0]이 "401"이면, 그런 파일은 존재하지 않는다는 메시지를 출력한다.

2-3. tmp[0]이 "500"이면, file name과 함께 GET을 사용하라는 메시지를 출력한다.

- 서버의 2번 이후 클라이언트의 1번이 실행된다.
- 클라이언트의 2-1의 1)이후 서버의 2-2의 1)이 실행된다.

4. PUT

-Server

1. inFromServer.readLine()을 통해 서버로부터 파일의 크기를 받아와 fileSize에 저장한다.

2-1. fileSize가 -1이면 status code 401을 포함한 response 메세지를 보내고 break한다.

2-2. fileSize가 -2이면 status code 500을 포함한 response 메시지를 보내고 break한다.

2-3. 나머지 경우, status code가 200인 response 메시지를 보낸다.

1) path에 현재디렉토리 경로 + 받아올 파일 이름을 저장한다.

2) data channel을 위한 ServerSocket의 accept 메소드를 통해 클라이언트의 접속을 기다린다.
- command line argument로 주어진 port number(주어지지 않은 경우 2121)을 이용해 data channel을 위한 ServerSocekt을 만든다.

3) accept결과 리턴된 socket에 대한 input stream인 dataFromClient와 path의 경로의 파일에 대한 output stream인 dataToFile을 만든다.

4) 5byte 크기의 byte array, header와 1000byte 크기의 byte array, data, 1005byte크기의 byte array, received_data를 만든다.

5) dataFromClient.read(received_data)를 이용해 클라이언트에서 보낸 데이터 메시지를 받아온다.

6) System.arraycopy 메소드를 이용해 received_data의 내용을 header와 data로 나눠서 저장한다.

7) dataToFile.write 메소드를 이용해 data에 있는 내용을 파일에 써준다.

8) cntSize에 data의 크기를 더해준다.

9) cntSize가 fileSize와 같아질 때까지 5)~8)과정을 반복해준다.

-Client

1-1. 서버로 전송할 파일을 입력하지 않은 경우(PUT만 입력한 경우), 서버에 file크기로 -2를 보낸다.

1-2.

1) 서버로 전송할 파일의 이름으로 File 타입 객체를 만들어 file에 저장한다.

2-1) 해당 파일이 존재하지 않는 경우 서버에 파일 크기로 -1을 보낸다.

2-2) 해당 파일이 존재하는 경우, File class의 length 메소드를 이용해 파일의 크기를 서버로 보낸다.

2. inFromServer.readLine()을 통해 서버로부터 response를 받아온 뒤, 이를 " "를 기준으로 split해서 tmp에 저장한다.

3-1. tmp[0]이 "200"인 경우

1) command line argument로 주어진 server host(주어지지 않은 경우 127.0.0.1)과 port number (주어지지 않은 경우 2121)을 이용해 서버와 연결된 Socekt을 만든다.

2) 위의 소켓에 대한 output stream인 dataToServer와 file에 대한 intput stream인 dataFromFile 을 만든다.

3) 5byte 크기의 byte array, header와 1000byte 크기의 byte array, data, 1005byte크기의 byte array, send_data를 만든다.

4) dataFromFile.read(data)를 통해 파일에 있는 테이터를 1000byte씩 읽어온다.

5) System.arraycopy 메소드를 이용해 header와 data의 내용을 send_data로 합쳐준다.

6) dataToClient.write를 이용해 send_data를 클라이언트로 전송해준다.

7) 파일을 모두 읽어서 클라이언트로 보낼 때까지 4)~6) 과정을 반복한다.

3-2. tmp[0]이 "401"인 경우 그런 파일이 (클라이언트에) 존재하지 않는다는 메시지를 출력한다.

3-3. tmp[0]이 "500"인 경우 file name과 함께 PUT을 사용하라는 메시지를 출력한다.

3-4. tmp[0]이 "502"인 경우 알 수 없는 이유로 실패했다는 메시지를 출력한다.

- 클라이언트의 1번 이후 서버의 1번이 실행되고, 서버의 2번 이후 클라이언트의 2번이 수행된다.
- 클라이언트의 3-1의 1)이후 서버의 2-3의 3)이 수행된다.

5. CD

-Server

1-1. 경로가 없는 경우 (CD만 입력한 경우) status code 200과 현재디렉토리를 포함한 response를 보내고 break한다.

1-2. 경로가 있는 경우 path에 경로를 저장하고, 상대경로인 경우, changeToAbsolute 메소드를 이용해 절대경로로 바꿔준다.

1-1) 만약 해당 경로가 디렉토리가 아니라면 status code가 501인 response를 보내고 break한다.

1-2) 현재디렉토리를 저장하고 있는 변수 currentDirectory에 path를 저장한 뒤 status code 200과 변경된 현재디렉토리를 포함한 response를 보내고 break한다.

-Client

1. inFromServer.readLine()을 통해 서버로부터 response를 받아온 뒤, 이를 " "를 기준으로 split해서 tmp에 저장한다.

2-1. tmp[0]이 "200"인 경우 tmp[3](변경된 경로)를 출력한다.

2-2. tmp[0]이 "501"인 경우 디렉토리명이 옳지 않다는 메시지를 출력한다.

- 서버의 1번 수행이후 클라이언트의 1번이 수행된다.

6. QUIT

-서버와 클라이언트 모두 exit를 1증가시켜서 while문을 빠져나간다.

7. status code

- status code는 아래와 같이 정의한다.
- 200 : 클라이언트의 명령을 정상적으로 수행한 경우
- 401 : 존재하지 않는 파일을 입력받아 실패한 경우
- 500 : 인자의 개수가 부족해 실패한 경우
- 501 : 존재하지 않는 디렉토리명을 입력받아 실패한 경우
- 502 : 알 수 없는 이유로 실패한 경우

<컴파일 및 실행 방법>

1. 컴파일

1) *cd 디렉토리경로*
   - 위의 명령을 이용해 FTPServer.java와 FTPClient.java가 있는 디렉토리로 이동한다.
   - ex)

```
C:₩Users₩tjtld>cd cnet
```

2) *javac FTPClient.java*
   - 위의 명령을 실행하면 FTPClient.java가 컴파일 되어 FTPClient.class가 생성된다.
   - ex)

```
C:₩Users₩tjtld₩cnet>ls
FTPClient.java  FTPServer.java

C:₩Users₩tjtld₩cnet>javac FTPClient.java

C:₩Users₩tjtld₩cnet>ls
FTPClient.class  FTPClient.java  FTPServer.java
```

3) *javac FTPServer.java*
   - 위의 명령을 실행하면 FTPServer.java가 컴파일되어 FTPServer.class가 생성된다.
   - ex)

```
C:₩Users₩tjtld₩cnet>ls
FTPClient.class  FTPClient.java  FTPServer.java

C:₩Users₩tjtld₩cnet>javac FTPServer.java

C:₩Users₩tjtld₩cnet>ls
FTPClient.class  FTPClient.java  FTPServer.class  FTPServer.java
```

2. 실행

1) *java FTPServer*

- 위 명령을 실행하면 FTPServer가 실행된다.
- 이때 Server에서는 명령어 채널을 위한 ServerSocket을 생성하고 accept()로 Client의 접속을 기다린다.

2) *java FTPClient*
- 위 명령을 실행하면 FTPClient가 실행된다.
- Server host와 port number를 이용해 소켓을 생성하면 서버의 accept()가 종료되면서 소켓을 리턴한다.

++ 1)과 2)는 각각 다른 cmd 창에서 실행해야한다.

3) FTPClinet에서 아래 다섯 가지 명령 중 하나를 입력한다. ([]부분은 필수는 아니다.)

1. *CD [이동할 디렉토리명(경로)]*
2. *LIST [디렉토리명(경로)]*
3. *GET [파일명(경로)]*
4. *PUT [파일명]*
5. *QUIT*

ex) 다음은 클라이언트에 CD, LIST, GET, PUT, QUIT를 입력한 예시이다.

- 초록색 화살표 : 클라이언트 -> 서버
- 파란색 화살표 : 서버 -> 클라이언트
- 노란색 화살표 : 서버와 클라이언트 사이의 데이터 전송