Program Design Description

Yang Meng 01679623
COMP 5630 Program Ass 1

I have implemented two Java programs, an HTTP client and a server. Both of them run HTTP/1.0 to extend the client and server to make use of some in-class application level headers

By using TCP connections, the HTTP Server creates a socket called “ss” for the local host on 127.0.0.1. When the server starts, it will go into a loop listening on the HTTP request from the client. And, if the client sends a request, the Server goes into another loop where it reads the bytes from the socket connection. When a GET or PUT request is received, the receipt of bytes is completed and the loop breaks. For a GET request, the end of request contains a set of \r\n.

The Server will process the request as soon as it receives the request. If it is a GET request, the data is written to the screen. Then they will be split to recreate the path name for the file that was requested. If the file exists, the status of “200 Found” is written to screen and the file at the path name is returned to the client. If there does not exist the file, status “404 Not Found” will be returned to the client.

As for a PUT request, the data string will be converted to bytes. And the array will be alter the size. The file is created from bytes and written to the server. The server will check whether the file that is PUT on the server exists. If it dose, a status code will be sent to the client. Besides, the file will be sent to the client as well.

When the data transmission is done, socket connection will be shutdown, and then closed. Then it will be waiting for the user to close the server socket.

The HTTP Client takes input from the console. Input consists of the host name, port number, request variable, and file name string. Depending on the request type, either the Client GET or PUT method is called. Both of them will return a string of bytes in the results variables.

As for a PUT request, host name, port number and file name are used as input. All if them will converted to bytes. A connection socket is created with the server based on the host name and port number. If the connection successfully set up, a byte array called buffer is created from the file and send to the server. Then the connection wait for the reply back from the server that is collected in the page string.

For a GET request, the host name, port number and file name are used as input and converted to bytes. Just like the PUT request, a connection socket is created with the server based on the host name and port number. If the connection socket set up successfully, connection socket will send a byte array containing the request and receive the bytes from the server. The bytes are collected in the page string which is printed to screen on the client.

Tradeoffs of design are considered. They are made include creating two classes, one for the server and another for the client, and stored in a single file. But a consideration is that server

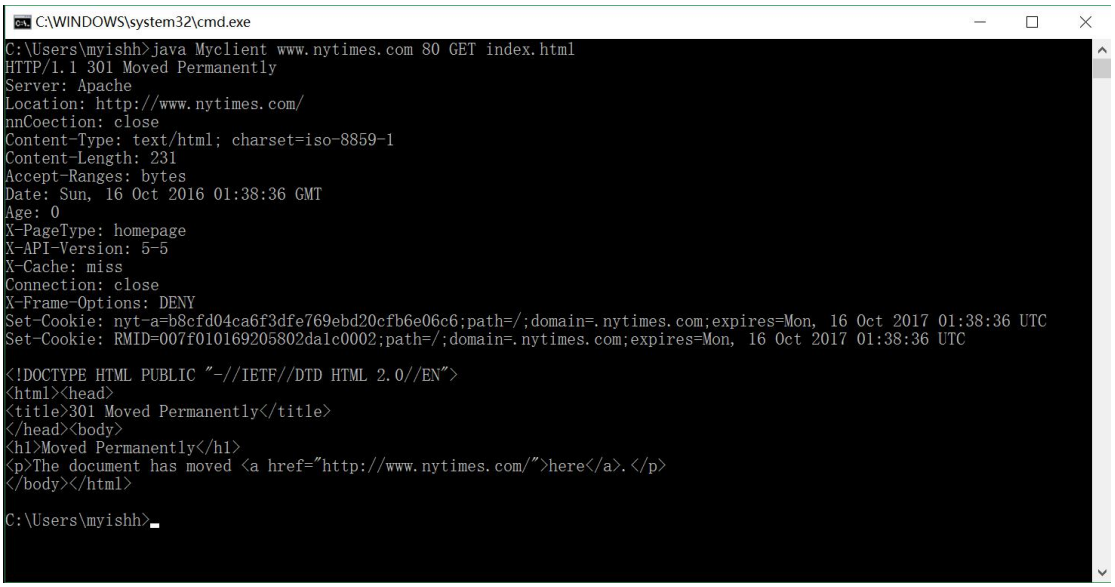
should be started and run separately from the client. This idea was abandoned and two separate projects were created and used.

As mentioned in the extra credit portion, possible improvements and extensions of program would be implementing the concurrency. In addition, another improvement is, my program could be more friendly to the careless people. That is to say, PUT & GET command could be case insensitive. Users could type in to PUT, Put or put to give the command, all the these command should be accepted and executed by the program.

Test Case

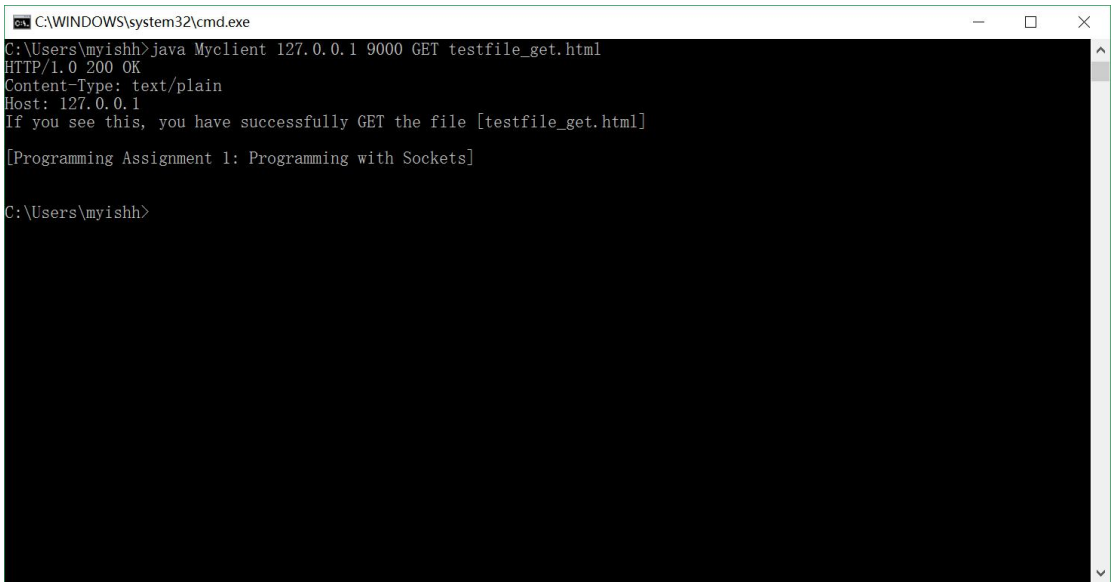
GET test case on a real website(www.nytimes.com)

Myclient www.nytimes.com 80 GET index.html

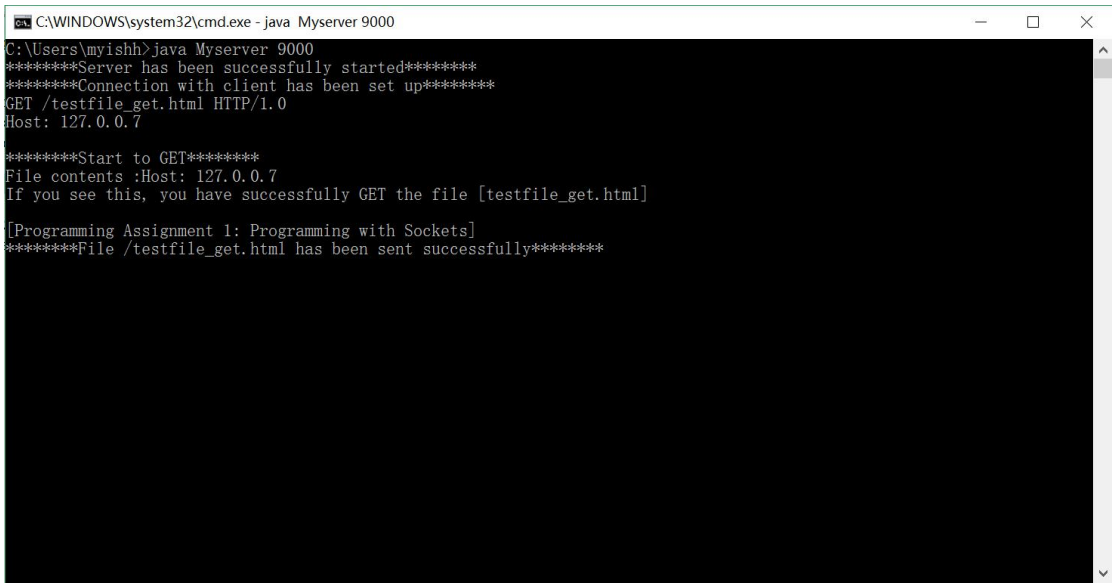


GET Test Case:

Client: Myclient 127.0.0.1 9000 GET testfile_get.html

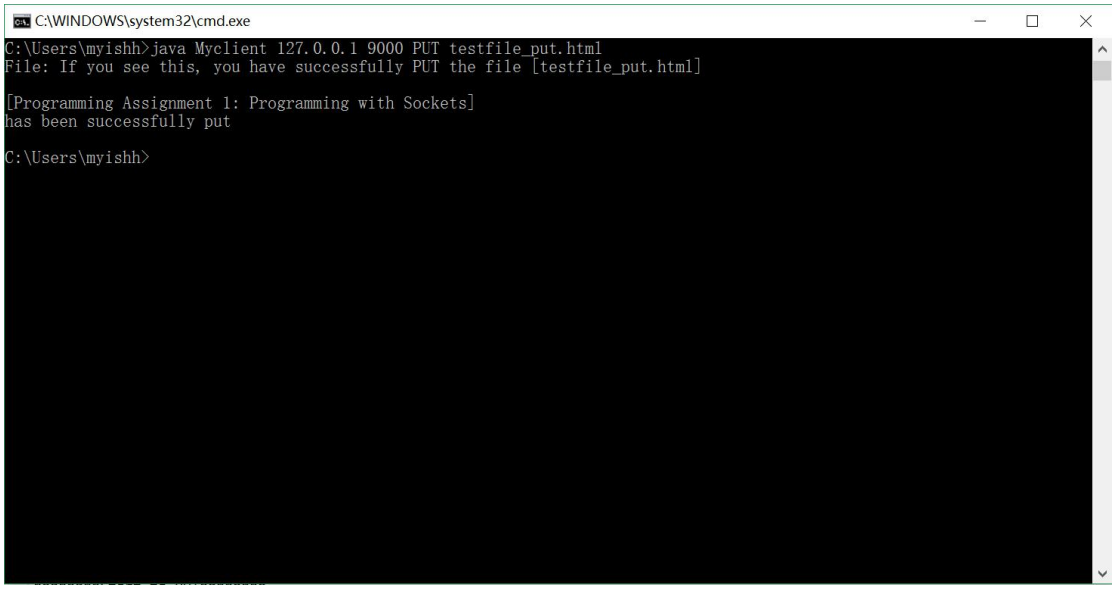


Server: Myserver 9000

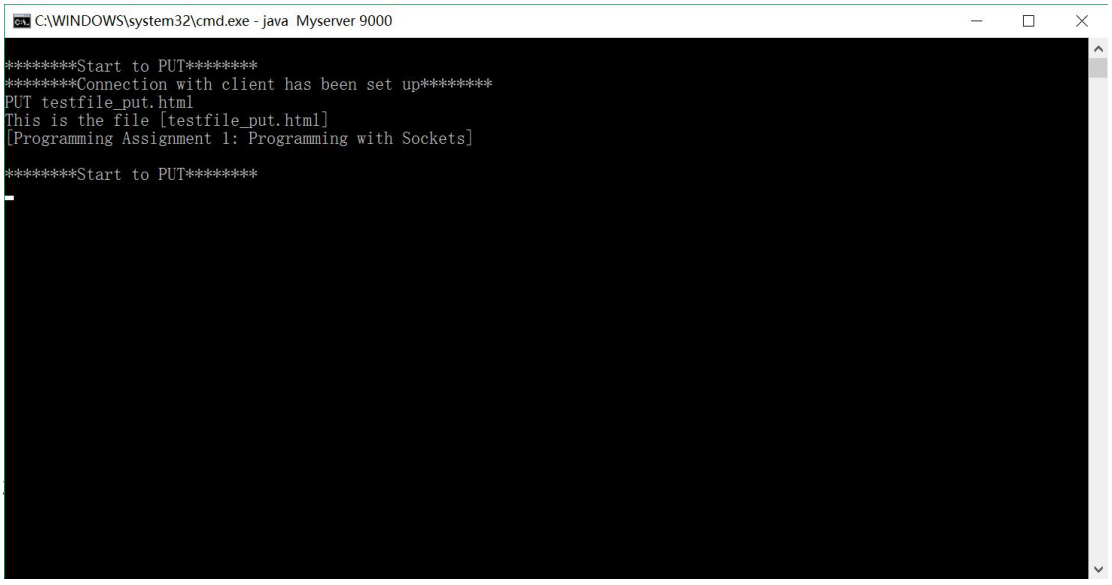


PUT test case

Client: Myclient 127.0.0.1 9000 PUT testfile_put.html



Server:



Attached file

COMP5630_ProgramAss1_MengYang.zip

- bin(Myclient.class/Myserver.class)
- Container(testfile_get.html)
- src(Myclient.java/Myserver.java)
- testfile_put.html