Building a Client/Server System Using Sockets

In this assignment you will extend your parsing and file-listing program from Homework 1 to operate over a network. Specifically, you will split your Homework 1 solution into two programs; one of which will be a client program and the other will be a server program. The server program will behave much as the original Homework 1 program did — it will read lines of input, identify valid HTTP GET requests and attempt to process all valid requests. However, whereas Homework 1 performed all I/O to standard input and standard output, your server program will perform all I/O to a TCP socket. The client program will interact with the user reading a line of input from standard input, send the line to the server program over a socket connection, receive a response line(s) from the server, echo all output from the server to standard output, and then wait for another line of input from the user. A separate new socket should be created for each interaction between the client and the server.

Remember to correct all errors from the HW 1 code before starting this homework — there will be significant grade penalty if any of those errors are not corrected. But copy your HW1 code to the HW3 directory before making any changes (do not modify the files in the HW1 subdirectory).

The figure below illustrates the desired conceptual operation of the client/server system in relation to operation of a prototypical solution to Homework 1. (Note that this is only meant to be an example of a possible solution. It is completely acceptable if the organization of your Homework 1 solution and your final client/server solution differs from mine.)
The server program, like most servers, will conceptually never terminate. It will be terminated by some external means such typing control-C in the shell. The client program will terminate when end-of-file is reached on standard input in exactly the same manner as in your solution to Homework 1.

The client and server programs will execute on different computers. The server program should be written to execute on the machine classroom.cs.unc.edu. It should further be written to listen for connections on port number 6000 + the last four digits of your social security number. This port assignment scheme should eliminate conflicts with well-defined port numbers and ensure students’ programs do not conflict with each other.

In order for your client/server system be correct, it should be the case that, modulo socket error messages (see below), the output of your client program (i.e., what is written to standard output) is indistinguishable from the output of a (correct) solution to Homework 1 when given the same input(s). Your server program should perform all of its HW1 related I/O to a socket. In particular, it should not interact with the user directly in any way. Specifically, it should not output prompts or other messages (except as noted below) and it should not read commands or other inputs from the keyboard or a file.

Both your client and server programs should be prepared to deal with socket related errors and attempt to recover from the errors when possible. Your program should be able to handle basic socket errors such as one side of the connection closing the connection unexpectedly or a client being unable to connect to a server. If a client or server encounters a socket error then it should print a meaningful error message to standard output. If the client or server is unable to recover from the socket error then it should terminate.

Grading

Again, remember to correct all errors from the HW 1 code before starting this homework – there will be significant grade penalty if any of those errors are not corrected. The detailed grading rubric and sample test input/output files will be shared by the TAs on piazza. Make sure your programs comply with all of the sample test cases.

To submit your program for grading, follow the general submission guidelines distributed with Homework 1 and send mail to the TAs when your program is ready. (Remember to give your CS Department UNIX id if you send mail from another account and use the subject line “COMP 431 HW3 ready” in your message.) You should have (at least) two Java files in your COMP431/submissions/HW3 directory. One program should be named Client.java and the other should be named Server.java. Please include copies of all .java and .class files used in this assignment in your submissions subdirectory.

As before, your programs should be neatly formatted and well documented. In general, 80% of your grade for a program will be for correctness, 20% for programming style and documentation. Refer to the handout on programming style and documentation for guidelines.