CSE 109: Systems Programming

Fall 2017

Program 8: **Due on Friday, December 9th at 9pm on CourseSite.**

Collaboration Reminder:

1. You must submit your own work.
2. In particular, you may not:
   (a) Show your code to any of your classmates
   (b) Look at or copy anyone else's code
   (c) Copy material found on the internet
   (d) Work together on an assignment

Assignment: Preparation

1. Make a *Prog8* directory in your class folder.
2. You will be writing a *Server.cpp* (and possibly *Server.h*) file for this assignment.
3. All source code files must have the comment block as shown at the end of this document. All files must be contained in your *Prog8* directory.
4. You are expected to check return values (except for printf/scanf and their variants) for errors via errno and report them (usually aborting when this happens).

Assignment:

In this assignment, you will be creating a basic network server and handling requests. We will refer to the components of this assignment Server and Client. The executables should end up in the same directory. The following page has an example of some working Client/Server code: http://math.msu.su/∼vvb/Java/samples/Invert/Invert.html be aware that you may need to check errors that it does not check and that the code is a bit sloppy. You can adapt portions of the code there for our purposes. You

will be expected to understand what functions you end up using and what some of the arguments are what they are.

Here is an additional link that goes into more detail if you are interested: http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html.

1. Server.cpp
   (a) This includes main and invokes the server.
   (b) When dealing with the network, you will need to use binary I/O.
   (c) Main will accept one command line argument (fail otherwise: return 1). This argument is the name of the file that we will be serving to the Clients.
   (d) Server will open up a port in the range [10000,10500].
      i. If any of those ports are available, it must use one of them.
      ii. If none of them are available, fail and return 2.
      iii. Upon success, create the file *connection.txt*. Write into this file the 2 bytes that represent the port number, followed by the 8 bytes that represent the size of the hostname, followed by the hostname (not null terminated), followed by the 8 bytes that represents the size of the filename that is being served, followed by the filename (not null terminated).

      The hostname is the machine you are currently on. This allows us to use *connection.txt* to find and connect to your Server instance. Lookup the *gethostname* function.
      iv. Note that this may change each time you run the program.
   (e) The Server will run until it receives a request for 0 bytes at starting location 0 (fulfill that request but do not accept any more requests).
   (f) When a Client connects they will send a request that contains at least 24 bytes but could be larger (up to 100 million). The first 8 bytes will indicate the total size of the incoming data.
   (g) Bytes 8-15 represent that starting location of the request.
   (h) Bytes 16-23 represent the amount of data being requested.
   (i) The remaining bytes are for a *integrity check*. Take the **sha1** hash of the remaining bytes, the resulting string will be our *checksum*.

   Unlike in Program 7, you will actually use the original data generated by the **sha1** hash and send that.

(j) Using starting location and the amount of data, read that data from the file we are serving. Provide only what is available. You may have to provide less than what was asked for.

(k) When responding to the Client: The first 8 bytes indicate how much data you are sending. The next 8 bytes indicate the size of the *checksum*, followed by the *checksum* (not null terminated). The remaining bytes, if any, are the requested data, if any, from the file.

2. Client

   (a) A Client executable will be provided but no source code is provided.

   (b) The Client will read the *connection.txt* file to determine how to connect to the server.

   (c) The Client and Server must be run from the same folder.

   (d) The Client, or multiple Clients, will make requests to the Server. They will make one request per connection. You do not have to handle multiple transactions per connection aside from **short count** handling.

   (e) The format of the Client message is described in the Server section. Note that the verification data is randomishly generated. The response from the Server is also described there.

   (f) The Client will verify both the *checksum* and the received data.

   (g) Some test cases for the Client are provided.

   (h) You may build your own Client but are not required to do so.

Testing:

1. Some test files are provided in the *p8student* directory.

   (a) *Client* and *ClientDebug* are both test Clients you can use.

   (b) The executables perform the same thing but one gives you diagnostic information that may be helpful.

   (c) #.input: Are input files that can be provided to the test Client.

   (d) server.txt: A source file that can be used by the Server. Make sure to test your Server using non-text files!

Submission:

1. Once ready to submit, you can package up the assignment as a .tgz file

   tar -czvf Prog8.tgz Prog8

   You must use this command in the directory that contains the *Prog8* folder, not within the directory.

2. Transfer *Prog8.tgz* to the Program 8 submission area of CourseSite.

Comment Block:

```
/*
    CSE 109: Fall 2017
    <Your Name>
    <Your user id (Email ID)>
    <Program Description>
    Program #8
*/
```