Group Project 1: Warm up of Interprocess Communication

CECS 326 – Operating Systems

You should submit the required deliverable materials on BeachBoard by **11:55pm, September 29th (Thursday), 2022.** We provide sample code files that you can complete the required functions.

1. Description

Question 1:

Sockets lecture describes certain port numbers as being well known—that is, they provide standard services. Port 17 is known as the quote-of-the-day service. When a client connects to port 17 on a server, the server responds with a quote for that day.

Modify the date server shown in Figure 1 so that it delivers a quote of the day rather than the current date. The quotes should be printable ASCII characters and should contain fewer than 512 characters, although multiple lines are allowed. Since these well-known ports are reserved and therefore unavailable, have your server listen to port 6017. The date client can be used to read the quotes returned by your server.

```
import java.net.*;
import java.io.*;

public class DateServer
{
  public static void main(String[] args) {
    try {
      ServerSocket sock = new ServerSocket(6013);

      /* now listen for connections */
      while (true) {
        Socket client = sock.accept();

        PrintWriter pout = new
         PrintWriter(client.getOutputStream(), true);

        /* write the Date to the socket */
        pout.println(new java.util.Date().toString());

        /* close the socket and resume */
        /* listening for connections */
        client.close();
      }
  }
  catch (IOException ioe) {
    System.err.println(ioe);
  }
}
}
```

Figure 1. Data Server

Question 2:

An echo server echoes back whatever it receives from a client. For example, if a client sends the server the string "Hello there! I am the client.", the server will respond with "Hello there! I am the server."

Write an echo server (you can use the Java networking API). This server will wait for a client connection using the *accept()* method. When a client connection is received, the server will loop, performing the following steps:

1. Read data from the socket into a buffer.
2. Write the contents of the buffer back to the client.

The server will break out of the loop only when it has determined that the client has closed the connection.

Hints: the date server of Figure 1 uses the *java.io.BufferedReader* class. *BufferedReader* extends the *java.io.Reader* class, which is used for reading character streams. However, the echo server cannot guarantee that it will read characters from clients; it may receive binary data as well. The class *java.io.InputStream* deals with data at the byte level rather than the character level. Thus, your echo server must use an object that extends *java.io.InputStream*. The *read()* method in the *java.io.InputStream* class returns −1 when the client has closed its end of the socket connection.

3: The Required Deliverable Materials

(1) A README file, which describes how we can compile and run your code.
(2) Your source code, you may use a Makefile (for C) and be submitted in the required format.
(3) Your short report, which discusses the design of your program.
(4) A recorded video shows the output and runtime

3. Submission Requirements

You need to strictly follow the instructions listed below:

1) This is a **group project**, please submit a .zip/.rar file that contains all files, only one submission from one group.

2) Make a **video** to record your code execution and outputs. The video should present your name or time as identification (You are suggested to upload the video to YouTube and put the link into your report).

3) The submission should include your **source code** and **project report**. Do not submit your binary code. Project report should contain your groupmates name and ID.

4) Your code must **be able to compile**; otherwise, you will receive a grade of zero.

5) Your code should not produce anything else other than the required information in the output file.

7) If you code is **partially completed**, please explain the details in the report what has been completed and the status of the missing parts, we will grade it based on the entire performance.

8) Provide **sufficient comments** in your code to help the TA understand your code. This is important for you to get at least partial credit in case your submitted code does not work properly.

Grading criteria:

Details	Points
Have a README file shows how to compile and test your submission	5 pts
Submitted code has proper comments to show the design	15 pts
Screen a ***video*** to record code execution and outputs	10 pts
Have a **report** (pdf or word) file explains the details of your entire design	20 pts
Report contains clearly individual contributions of your group mates	5 pts
Code can be compiled and shows correct outputs	45 pts

4. Policies

1) Late submissions will be graded based on our policy discussed in the course syllabus.
2) Code-level discussion is **prohibited**. We will use anti-plagiarism tools to detect violations of this policy.