HÁSKÓLI ÍSLANDS
Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild

TÖL401G: Stýrikerfi / Operating Systems · Vormisseri 2022

Assignments 7–8 · To be solved until 15.2.2022, 13:00

After assignment submission deadline 15. February 2022, 4 (out of 8) assignment solutions need to have been submitted – Otherwise, you run the risk of being unregistered from this course!

Assignment 7

Create a Java solution for the producer-consumer problem with a zero-capacity buffer, i.e. use connection-less unreliable socket communication (this means: two communicating processes running at the same time, each having its own `main` method, and are thus started in two separate Java virtual machines – just as illustrated in chapter 3 example demo videos):

- One Java program shall serve as the producer process that sends items to the consumer. As data it shall produce strings that contain consecutive increasing numbers, i.e. the first produced data item is "1", the second "2", and so on. For debugging purposes, let the producer print to `System.out` the produced item (in addition to sending it).
- Another Java program shall serve as the consumer process. As processing of the received data, it shall print the received data item to `System.out`.

Hints:

- Use reasonable class names for implementing producer and consumer so that it is clear which Java file contains the producer main method and which contains the consumer main method – names such as *Client* or *Server* are no reasonable names in the context of the producer-consumer problem – rather use, e.g., *Consumer* and *Producer*!
- As connection-less unreliable communication is used, produced data will get lost if it is sent from the producer to the consumer while the consumer is busy. Therefore, you need to take care that the producer only sends a new piece of data after the consumer has informed it via a special message that it is ready for consuming a new piece of data.

  You only have to care and deal with this aspect of connection-less unreliable communication, i.e. prevent overflow of the zero-capacity buffer – you can assume that otherwise, messages do not get lost. (Due to unreliable communication, messages may in fact get lost anytime. But as long as you run producer and consumer on the same machine, this will not happen as long as you do not overflow zero capacity buffers.)
- To turn an `int i` into a string: `Integer.toString(i)`. To get a byte array representation (e.g. to be used in a datagram) of a `String s`: `s.getBytes()`.
- If you want to run the two processes on your local machine without needing to know the IP address or hostname of your machine, you can use "localhost" as hostname. ("localhost" always refers to your local machine, independent from the real hostname or IP address of your machine.)

Assignment 8

1. Change the connection-oriented reliable stream communication Java server presented on the lecture slides (Section 3.5) so that it does not simply echo back, but interprets the string sent by the client as an integer n and calculates recursively Fibonacci number n and sends the result back as string.

   Hints: To turn a `String s` into an integer: `Integer.parseInt(s)`. To turn an `int i` into a string: `Integer.toString(i)`.

2. Turn your server into a multi-threaded[1] server that is able to process multiple Fibonacci number calculation requests from multiple clients at the same time.

3. If you have a POSIX-based system (Linux, Mac OS, etc.), start a shell console window and enter: `netstat -a`

   If you have Microsoft Windows, enter in the command prompt window: `netstat /a`

   On MS Windows, you start the command prompt as follows:

   - In older versions having the classic "Start" menu: open the command prompt window by entering `cmd` into the dialogue you get via *Start→Run* or open it directly via *Start→All Programs→Accessories*
   - In newer versions having the tile-based start menu, use the search facility (by clicking on the magnifying glass icon) and enter `cmd` and choose *Command Prompt*.

   (a) Do this while your server process is running without any client being connected.

   (b) Do this while your server process is running with one (or more) client being connected.

   In each of these two cases, find your server process in the netstat output (in the output for sockets of the Internet domain) and copy all lines that relate to your server into your `readme.txt` file!

4. Explain *briefly* what you can learn from the netstat output, i.e. what does the pair shown for "Local Address" and the pair shown for "Foreign Address" and the "State" mean?

General comments concerning Java assignment submission

For Java programming assignments, a special programming assignment Gradescope mode is used, hence **do not submit any PDF**, only the Java source code (`*.java` files plus a readme file) of your programming solutions – the compiled byte code (`*.class` files) is not required.

1. Use **zip format** to create a single file archive of the multiple source code files (no other file formats accepted)! If your source code needs be located in a folder, e.g. due to Java package structure, take care that this directory structure is contained in the zip file.

2. In addition, provide inside the zip file a short text file **readme.txt** that contains information how to start the involved processes (e.g. "To start client process: java ch3Processes.connectionOrientedSocket.ConnectionOrientedClient 42 localhost"). If it matters in which order the processes are started, mention this as well.

[1] If you do it right, you have just to change three lines in the example from Section 3.5 to achieve this.

3. For assignment 7, the Java source code and the readme.txt are sufficient.

4. For assignment 8, the Java source code resulting from part 1 and 2 (sufficient to provide the final version of part 2, because this includes essentially part 1) and the readme.txt need to be submitted; but you also need to answer parts 3 and 4: simply include the answers in file readme.txt.