V 1.0, 8th-Jan-2018

DSPro

Optional Practical on Distributed Systems

General guidelines

- Each exercise is worth 0,5 points out of a total of 1,5 points which is the credit assigned to the present homework in the course. You have no obligation to deliver the full homework assignment; any part thereof will count if properly explained and developed.
- Program sources must be clearly and lightly commented
- Apply a simple OO design strategy or structured programming strategy depending on whether you are programming in Java or in C, respectively
- Include rich explanations of your design decisions and the unit tests that demonstrate the correction of your programs
- You can only submit your original work, programs and explanations. You can incorporate source code from open software projects, in which case you must cite the authors and their overall weight must be small.
- Submit the solution to **each exercise** in a **separate folder** which name must be "Exercise 1", "Exercise 2", etc. Each folder must contain one folder for the sources (**src**), another folder for the explanations (**docs**) and a last folder for the executable files (**bin**)
- Include software construction instructions if necessary
- Compress the complete folder structure mentioned above in a .zip file (Please, use .zip exclusively, otherwise, I might not be able to decompress the archive which might hamper your grade)
- Submit the zip-compressed archive to foces.informatica.unileon@gmail.com by Monday 15th/Jan/2018 11:59:59 pm UTC

1. Simple probabilistic time synchronization algorithm

Using ICMP timestamps, build a C program that synchronizes its host's clock (The client) with another host of your choosing (The server). The client program will fetch the server's time several times and each time it will calculate the time difference (delta); finally, it will compute the mean delta and proceed to gradually adjust the local clock by calling the Linux adjtime() function.

a. The program must print out the following items:
   i. The delta obtained at each timestamp request
   ii. The mean delta
   iii. The local time before invoking adjtime()

iv. The local after invoking adjtime()

b. Explain what tests you will perform to demonstrate that the program functions correctly
c. Highlight the Linux commands involved in managing the local clock that you used to perform the tests

2. RFC 868 time protocol

Skim the RFC 868 time protocol and implement the following RFC 868 C/S pairs:

Implemented over	Client	Server
TCP	C	Java
UDP	Java	C
JRMP	Java RMI	Java RMI

a. Deliver each C/S pair in a separate folder which containing the corresponding sources, executable code and documentation
b. Provide extensive explanations of the considerations and problems that you found in this question
c. Explain the tests that you designed to check the code

3. RMI C/S

Write an RMI server that keeps the counter of certain events in a distributed system; these specific events are not relevant to this question. Your task consists of building a Server that implements two remote methods for updating the counter, one for incrementing it and the other one for decrementing it. The methods' signatures follow:

```
long int increment(); //Increments the counter (counter++)
long int decrement(); //Decrements the counter (counter--)
```

Upon server initialization, the counter must be initialized to 0.

a. Write an RMI client program that allows us to test the server
b. Provide an extensive discussion of your software design and the tests that demonstrate that it functions correctly
c. Highlight the core hurdles involved in this small distributed project