Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Priv.-Doz. Dr. Frank Huch, Marius Rasch

Institut für Informatik

2. Übung zur Vorlesung „Concurrent and Distributed Programming“

Abgabe am Dienstag, 23. April 2019 - 14:00

In this exercise sheet we start with programming in Erlang. To do corresponding exercises you should install Erlang on your computer.

You will find the Erlang binaries at https://www.erlang.org/downloads. For many Unix systems compiled packages are available, which can be installed using the corresponding package manager (e.g. `apt-get install erlang` on Debian/Ubuntu). As an alternative you can download and compile the Erlang source (for instance by using the Kerl script).

After installing Erlang you can start the Erlang Shell using the `erl` command. To compile a modul `example` you could then do the following:

[]erlang 1> c(example). ok,example

If this was successful you can call functions in this module:

[].erlang 2> example:fac(4). 24

To get started the documentation and the tutorial “Learn You Some Erlang for Great Good” are useful:

https://www.erlang.org/docs

http://learnyousomeerlang.com/

Aufgabe 1 - Counter with GUI — 1 Punkt

Expand your counter from exercise sheet 1 task 2 by a class `CounterGUI` to provide a graphical output. Every counter should be shown and controlled in a separate window. Therefore every window should provide the following buttons:

- Stop: the counter stops on the current value
- Start: the counter resumes counting at the same speed
- Close: closes the current window

Make sure that the program terminates, if no counter is shown anymore.

Add two new buttons to the window afterwards:

- Copy: Copies the counter with its current value and current speed. The new counter behaves independent of the old counter.
- Clone: Clones the counter. The clones behavior should be identical. If one counter is stopped for instance, the other counter should stop too.

Hint: The standard GUI package for Python is TkInter. If you are new to Python, we provide a simple TkInter GUI for you.

Aufgabe 2 - Expansion of Chan

1 Punkt

In the lecture the implementation of an unbounded buffer (`Chan`) was presented. This implementation should be expanded in the following.

1. Expand the implementation of `Chan` by the methods `add_multiple` and `un_get`. `add_multiple` should add elements of a Collection to the channel and `un_get` should add an element to the wrong end of the chan.
2. Are your implementations correct in every situation or do they sometimes behave unexpected? Sketch an idea how to improve this behavior if necessary.
3. Implement `Chan` in a way, that even `is_empty` and `un_get` behave as expected.
4. Modify the implementation given in the lecture in a way, that the buffer is bounded by a size specified during construction. A `MVar` would be a buffer with capacity one.
5. Implement a bounded buffer by using an array. Compare this solution with the previous one. Especially analyze how accurate the concurrent access is.

Aufgabe 3 - Messages in Erlang

1 Punkt

In this exercise you should write simple programs in Erlang to analyze the message based communication in Erlang.

1. Write a test program (as small as possible) illustrating the matching order of the `receive` expression.
2. Implement a counter process, which can be incremented by appropriate client processes. Program ten client processes, which increase the counter by 1000 in a concurrent way.

The counter program from the exercise