Threads in Clojure

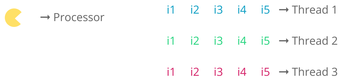
November 21, 2015

I was writing a post about Clojure state management, and I found out, there is one important topic I need to mention. Threads in Clojure. I came into contact with threads in Clojure, when I worked with quartzite library .
This time I found out, it's really interesting topic, and I hope, this post will help you understand threads, and you'll enjoy it.

What is a thread

You can imagine thread as a serie of instructions to execute. One program can spawn many threads, and one thread can have many instructions. If we have one thread, we talk about Single threading, and if we have more than one thread, we call it Multithreading

We can imagine program as a pacman game. Where pacman is a processor, pacman board is a program, one color is one thread and instead of dot is an instruction



Single threading
(Program with one thread)

If we have a system with one processor (single-processor system) and program has only one thread (one line of dots), processor (pacman) is executing (eating) instructions one after another.

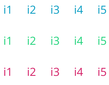
So the program is executing instructions like this :

If we have system with more processors (multiprocessor system) , and only one thread , the instructions in thread are executing by only one processor. The second processor is not used.

Multithreading with single processor system
(Program with multiple threads)

if our program have more threads, processor is executing instructions concurrently. The processor switches back and forth between threads. It is important to know, the are no guarantees about switching back and forth between threads (When the program execution will be handed from one thread to another).

The actuall thread execution can happen in multiple ways:



This program can be executed:

Possible execution scenario 1

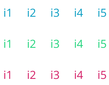
Possible execution scenario 2

> NOTE:
> Remember,there is no guarantee about how the instructions will be executed, so there are multiple possibilities how instructions can be executed, and those are only two of them. It means, the program is nondeterministic.

Multithreading with multiprocessor system
(Multiple threads on hardware with more than one processor)

Multiprocessor system allows the program to execute more than one instruction at the same time. System with two processors can execute two instruction synchronously.

Example with two processors



This example can be executed:

How you can start new thread in Clojure

In Clojure you can create new thread using java class `Thread.` - by calling (`.Thread (my-fn)`). You need to give to thread a function without arguments. Then you can start thread using `.start`

I will show you simple example of starting new thread.

We have functon named `make-fn` . This funtion takes one argument `text` and as a result returns an anonymous function without an argument. This anonymous function is looping function, and will print numbers from 1 - 10, prepended by `text` (argument from the outer function - as you can see, we are uilizing closures).

If we will call `make-fn` function with `"from-thread"` as an argument, function returns:

We define new function `run-in-two-threads!`. This function will set our looping function to be executed in two different threads. We call `make-fn` function with argument "thread-1" for the first thread, and for the second thread we call `make-fn` with an argument "thread-2"

Calling `run-in-two-threads!` function:

This is only simple explanation of threading principles. But threading is big topic, because there are a lot of ways how you can manage threads, processors ...
But I hope, this post helped you to understand basic principles, and if you want to know something more, or you think there is something important I forgot to mention, feel free to write a comment.