

# Functions with parallel execution and communication channel

Team: Alex (ajc2) e Victor (vlo2)



# Motivation

- Introduce parallelism in functional language 3:
  - Raising performance of the language.
  - Better use of processing power.
  - Enable sharing of data and resources.
  - Division of work.
  - Allow reuse of services already available.

# Motivation

- Developed by Ericsson in 1986.
- Creating and managing processes is a trivial task in Erlang.



# Motivation

- Competition happens between lightweight processes.
- There is a data type called PID, the parallel process identifier and in which messages can be sent.



# Erlang - Example 01

```
loop() ->  
  receive  
    "Hi!" ->  
      io:format("Hello world!!!"),  
      loop();  
    Any ->  
      io:format("Received: ~p~n", [Any]),  
      loop()  
  end.
```



# Erlang - Example 02

```
defmodule Ping do
  def start do
    loop()
  end

  def loop do
    receive do
      {:pong, from} ->
        IO.puts "ping ->"
        :timer.sleep 500
        send(from, {:ping, self})
      {:ping, from} ->
        IO.puts "<- pong"
        timer.sleep 500
        send(from, {from, {:pong, self}})
    end
  end
end
```



# Goals

Extend the functional language 3 to include competition, based on the Erlang programming language.

LF3



# Language - LF3

- Extends Functional Language 2 with list and list comprehension.
- A List is a value.
- A program is an expression.



# BNF

ExpUnaria ::= "-" Expressao | "not" Expressao | "length" Expressao  
| head(Expressao) | tail(Expressao)  
| ExpCompreensaoLista  
| **wait(ValorInteiro)**  
| **receive(ValorString "," ValorInteiro)**  
| **send(ValorString "," ValorString)**

**DecProcesso ::= "process" Id ExpDeclaracao "end"**

## Link repositório:

<https://github.com/victorlaerte/cin-plp-project/blob/master/BNF.md>

# Sintax - creating a process

```
process idProcess1
```

```
  let fun functionExample = wait(3) // This will lock the process by 3 seconds
```

```
    send("main", "message string example")
```

```
    // This will send message to the 'main' process
```

```
  in functionExample()
```

```
end
```

# Sintax - creating a process

```
process idProcess2
```

```
    let fun teste msg1= msg1 in
```

```
        receive(idProcess1, 60) // This will wait until process1 response or 60 seconds timeout
```

```
end
```

# Sintax - ping pong example

```
process main
```

```
  let fun teste msg1 msg2 = msg1+msg2 in
```

```
    receive(idProcess1, 60) // This will wait until process1 response or 60 seconds timeout
```

```
    receive(idProcess2, 60) // This will wait until process2 response or 60 seconds timeout
```

```
end
```

```
process idProcess1
```

```
  let fun ping = wait(3) // This will lock the processe by 3 seconds
```

```
    send("main", "ping") // This will send message 'ping' to the mainProcess
```

```
  in ping()
```

```
end
```

```
process idProcess2
```


```
  let fun pong = wait(5) // This will lock the processe by 5 seconds
```




```
    send("main", " pong") // This will send message 'pong' to the mainProcess
```



```
  in pong()
```



```
end
```

# General Task Board



 victorlaerte / cin-plp-project > Projects > Functions with parallel execution and communication channel



 Show menu  Add cards  Exit fullscreen

TODO 1  



 Release new feature and create final presentation 

Added by victorlaerte



DOING 4  

 Update the BNF model 



Added by victorlaerte

 Present to the professor or monitor a proposal of how the functionality syntax would look like and analyze the suggestions 



Added by victorlaerte



 Start development and tests 

Added by victorlaerte



 Create presentation 



Added by victorlaerte

VALIDATING OR TESTING 1  

 Identify which mechanisms we can use in Java to build this functionality (Like BlockingQueue) 


Added by victorlaerte

DONE 1  

 Study functional languages (Like F#) to identify mechanisms of communication to understand how it will works 

Added by victorlaerte

# Dev Task Board

 victorlaerte / cin-plp-project > Projects > Dev Taskboard

Show menuAdd cardsExit fullscreen

TODO5

Create process declaration

Added by victorlaerte

add parallel function evaluation when declared as process

Added by victorlaerte

Create send function

Added by victorlaerte

Create receive function

Added by victorlaerte

Create documentation

Added by victorlaerte

DOING1

Create wait function

Added by victorlaerte

VALIDATING OR TESTING0

DONE0

# Repository



<https://github.com/victorlaerte/cin-plp-project>

# References

Martin Brown (10 de maio de 2011). «Introduction to programming in Erlang, Part 1: The basics» (em inglês). IBM Developer Works. Consultado em 01 de maio de 2017

Erlang (linguagem de programação). In: Wikipédia: a enciclopédia livre. Disponível em:  
<[https://pt.wikipedia.org/wiki/Erlang\\_\(linguagem\\_de\\_programa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/Erlang_(linguagem_de_programa%C3%A7%C3%A3o))> Acesso em: 29 abril 2017.