Erlang's Path: Concurrency to Seamless Distribution

Robert Fiko

Erlang 101 

Erlang 101
Math 😐

```
-module(basics).


add(X, Y) -> X + Y.
```

Erlang 101
Visibility 🧐

```
-module(basics).
-export([add/2]).

add(X, Y) -> X + Y.
```

Erlang 101
Clauses 🤷

```
-module(basics).
-export([add/2]).

add(X, Y) -> X + Y.

is_zero(0) -> true;
is_zero(_) -> false.
```

Erlang 101
Lists 

```
-module(basics).


hd([H|_]) -> H.
```

Erlang 101
Lists 

```erlang
-module(basics).

hd([H|_]) -> H.

% map
```

Erlang 101
Lists 

```erlang
-module(basics).

hd([H|_]) -> H.

% map
map(_, []) -> [];
```

Erlang 101
Lists 

```
-module(basics).


hd([H|_]) -> H.


% map
map(_, []) -> [];
map(F, [H|T]) -> [F(H)|map(F, T)].
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1>
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
2> List2 = lists:seq(1,10).
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
2> List2 = lists:seq(1,10).
[1,2,3,4,5,6,7,8,9,10]
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
2> List2 = lists:seq(1,10).
[1,2,3,4,5,6,7,8,9,10]
3> [ X * 2 || X <- List2 ].
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
2> List2 = lists:seq(1,10).
[1,2,3,4,5,6,7,8,9,10]
3> [ X * 2 || X <- List2 ].
[2,4,6,8,10,12,14,16,18,20]
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
2> List2 = lists:seq(1,10).
[1,2,3,4,5,6,7,8,9,10]
3> [ X * 2 || X <- List2 ].
[2,4,6,8,10,12,14,16,18,20]
4> [ X || X <- List2, X rem 2 == 0 ].
```

```
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1]
[jit]

Eshell V12.3.2  (abort with ^G)
1> List = [1,2,3].
[1,2,3]
2> List2 = lists:seq(1,10).
[1,2,3,4,5,6,7,8,9,10]
3> [ X * 2 || X <- List2 ].
[2,4,6,8,10,12,14,16,18,20]
4> [ X || X <- List2, X rem 2 == 0 ].
[2,4,6,8,10]
```

```
5> this_is_an_atom.
```

```
5> this_is_an_atom.
this_is_an_atom
```

```
5> this_is_an_atom.
this_is_an_atom
6> "this_is_string". % list of chars really...
```

```
5> this_is_an_atom.
this_is_an_atom
6> "this_is_string". % list of chars really...
"this_is_string"
```

```
5> this_is_an_atom.
this_is_an_atom
6> "this_is_string". % list of chars really...
"this_is_string"
7> Atoms = [pear, apple, mango].
```

```
5> this_is_an_atom.
this_is_an_atom
6> "this_is_string". % list of chars really...
"this_is_string"
7> Atoms = [pear, apple, mango].
[pear,apple,mango]
```

```
5> this_is_an_atom.
this_is_an_atom
6> "this_is_string". % list of chars really...
"this_is_string"
7> Atoms = [pear, apple, mango].
[pear,apple,mango]
8> [ {N, A} || N  <- List, A <- Atoms].
```

```
5> this_is_an_atom.
this_is_an_atom
6> "this_is_string". % list of chars really...
"this_is_string"
7> Atoms = [pear, apple, mango].
[pear,apple,mango]
8> [ {N, A} || N  <- List, A <- Atoms].
[{1,pear},
 {1,apple},
 {1,mango},
 {2,pear},
 {2,apple},
 {2,mango},
 {3,pear},
 {3,apple},
 {3,mango}]
```

```
10> {J, I} = {1,apple}.
```

```
10> {J, I} = {1,apple}.
{1,apple}
```

```
10> {J, I} = {1,apple}.
{1,apple}
11> I.
```

```
10> {J, I} = {1,apple}.
{1,apple}
11> I.
apple
```

```
10> {J, I} = {1,apple}.
{1,apple}
11> I.
apple
12> J.
```

```
10> {J, I} = {1,apple}.
{1,apple}
11> I.
apple
12> J.
1
```

Error handling and guards 💪🏻

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 ...
1>
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 ...
1> math:add(5, 3).
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 ...
1> math:add(5, 3).
8
2>
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 ...
1> math:add(5, 3).
8
2> math:add(alma, korte).
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) -> X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 ...
1> math:add(5, 3).
8
2> math:add(alma, korte).
** exception error: an error occurred when
evaluating an arithmetic expression
     in operator  +/2
        called as alma + korte
     in call from math:add/2 (math.erl, line
4)
3>
```

```
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
   X + Y.
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
   X + Y.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-
bit] [smp:12:12] [ds:12:12:10] [async-
threads:1] [jit]

Eshell V12.3.2  (abort with ^G)
1> math:add(alma, korte).
** exception error: no function clause
matching math:add(alma,korte) (math.erl,
line 4)
2>
```

```
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
   X + Y.

add(_, _) ->
  error.
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
    X + Y;

add(_, _) ->
  error.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-
bit] [smp:12:12] [ds:12:12:10] [async-
threads:1] [jit]

Eshell V12.3.2  (abort with ^G)
1> math:add(3, 2).
5
2> math:add(3, apple).
error
3>
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
   X + Y;

add(_, _) ->
  error.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-
bit] [smp:12:12] [ds:12:12:10] [async-
threads:1] [jit]

Eshell V12.3.2  (abort with ^G)
1> math:add(3, 2).
5 <- this is a value
2> math:add(3, apple).
error <- this is a value
3>
```

```
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
  {ok, X + Y};

add(_, _) ->
  {error, badarg}.
```

```erlang
-module(math).
-export([add/2]).

add(X, Y) when is_integer(X) and
is_integer(Y) ->
  {ok, X + Y};

add(_, _) ->
  {error, badarg}.
```

```
➜  erlang_demo erlc math.erl
➜  erlang_demo erl
Erlang/OTP 24 [erts-12.3.2] [source] [64-
bit] [smp:12:12] [ds:12:12:10] [async-
threads:1] [jit]

Eshell V12.3.2  (abort with ^G)
1> math:add(3, 2).
{ok,5}
2> math:add(3, apple).
{error,badarg}
3>
```

Erlang style error handling
vs.
Exceptions?

Concurrent Erlang 

Concurrent Erlang

Getting the current process's PID

```
SelfPid = self().
```

Concurrent Erlang

Getting the current process's PID

```
SelfPid = self().
```

Message sending

```
SelfPid ! hello.
```

Concurrent Erlang

Getting the current process's PID

```
SelfPid = self().
```

Message sending

```
SelfPid ! hello.
```

See what messages the Shell process got

```
flush().
```

Concurrent Erlang

Getting the current process's PID

```
SelfPid = self().
```

Message sending

```
SelfPid ! hello.
```

See what messages the Shell process got

```
flush().
```

Spawn new processes

```
spawn(fun () -> SelfPid ! "Hello world!" end).
```

Erlang Data Model 💾

- Each and every process has their own heap and stack
- Data is immutable
- Garbage collector
- Processes have message queue, used for inter-process communication

Task 1:
Ping Pong 🏓

Task description:

https://github.com/robertfiko/concurr_to_distr/blob/main/exercises/ping_pong.erl

Task 2: Fibonacci 🐚

Task description:

https://github.com/robertfiko/concurr__to__distr/blob/main/exercises/fibonacci.erl

Distributed Erlang 

Erlang node

- **epmd (Erlang Port Mapper Daemon)**
- **cookies**
- **sname, name**

Erlang nodes

- start two nodes:
  - `erl -sname … -setcookie biscuit`
  - `net_adm:ping`

They will talk! :)

Complex Task: Web crawler 🌐

We have a given list of URLs of Wikipedia articles.

We want to count the appearance of certain words.

Further description/instruction:

https://github.com/robertfiko/concurr_to_distr

Questions? 🥹

Thanks!