
M36FK
LM
311N

Life of an Erlang Process

@robertoaloi

Erlang
SOLUTIONS

Let's talk about Erlang

I won't talk about the Erlang syntax.

Syntax is Irrelevant

Programming Language is Not

When you'll look at the syntax, it'll be too late.

ERLANG HAS BEEN DESIGNED TO HELP YOU WRITING

SCALABLE

SYSTEMS

ERLANG HAS BEEN DESIGNED TO HELP YOU WRITING

FAULT-TOLERANT

SYSTEMS

ERLANG HAS BEEN DESIGNED TO HELP YOU WRITING

HIGHLY AVAILABLE

SYSTEMS

ERLANG HAS BEEN DESIGNED TO HELP YOU WRITING

MASSIVELY CONCURRENT

SYSTEMS

ERLANG HAS BEEN DESIGNED TO HELP YOU WRITING

DISTRIBUTED

SYSTEMS

ERLANG HAS BEEN DESIGNED TO HELP YOU WRITING

SOFT REAL-TIME

SYSTEMS

Scalable System	Tons of Users
Users as Processes	Tons of Processes

The Erlang Rationale

Think Erlang

Think Processes

Tons of them.

DEALING WITH TONS OF PROCESSES

Cheap to create

Cheap to context switch

OS processes out of discussion

Use lightweight processes

Built-in distribution to horizontally scale

SHARED MEMORY (or lack thereof)

Shared memory
can leave the system
in an inconsistent state
after a restart or crash.

Avoid it.

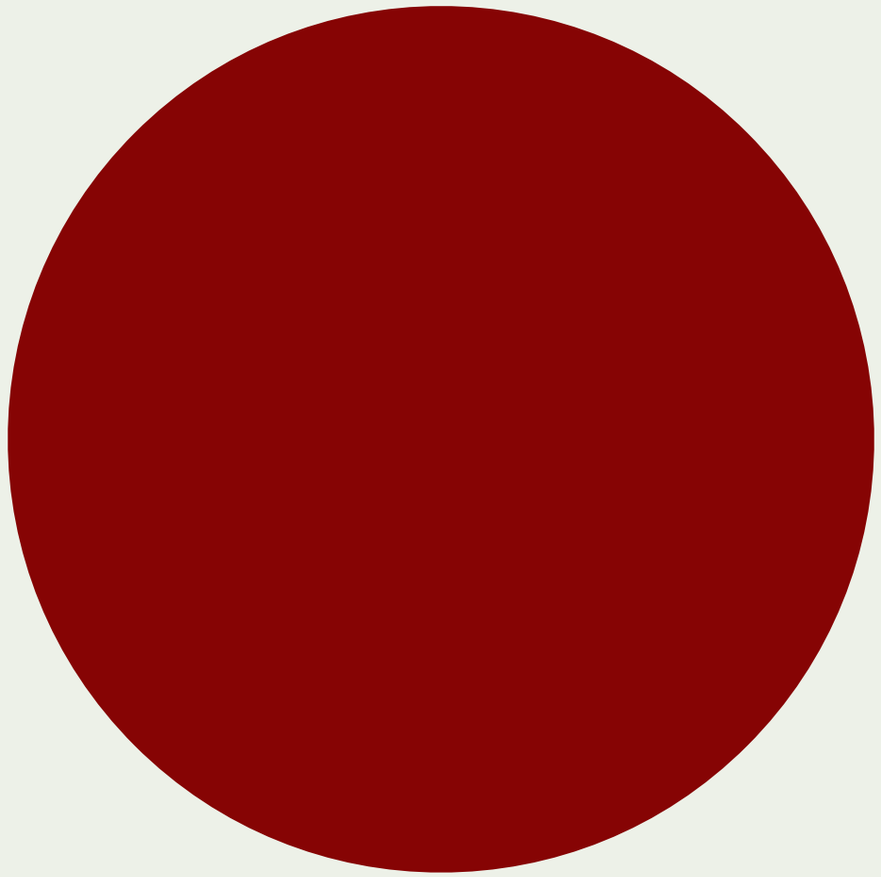
FAIL FAST

Faults are everywhere,
you cannot prevent them.

Report failure
and die.

Processes are good.

Can you show me one?

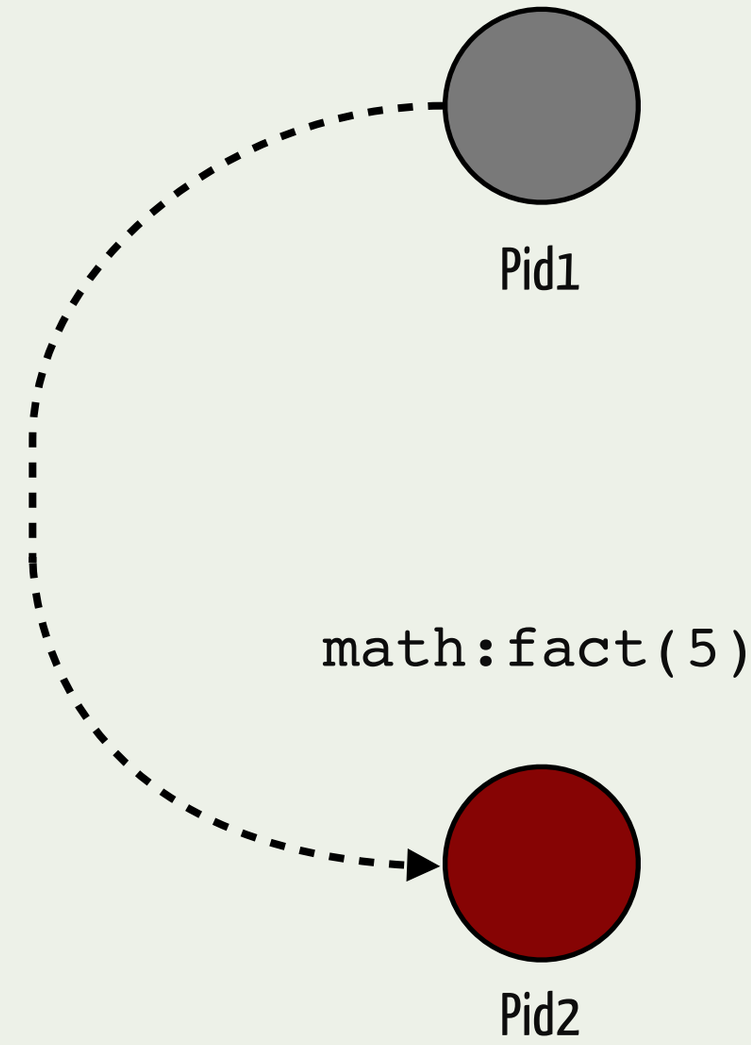


<0.42.0>

This is a little embarrassing, but...
Ahem...
I'm not sure I understand everything of it...
I've heard rumours, still...
Do you, do you mind if I ask you...

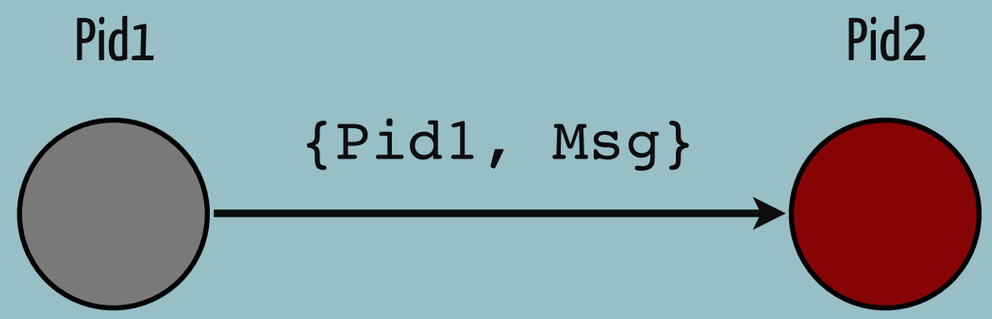
How are processes made?

```
spawn(math, fact, [5])
```



MESSAGE PASSING

Send

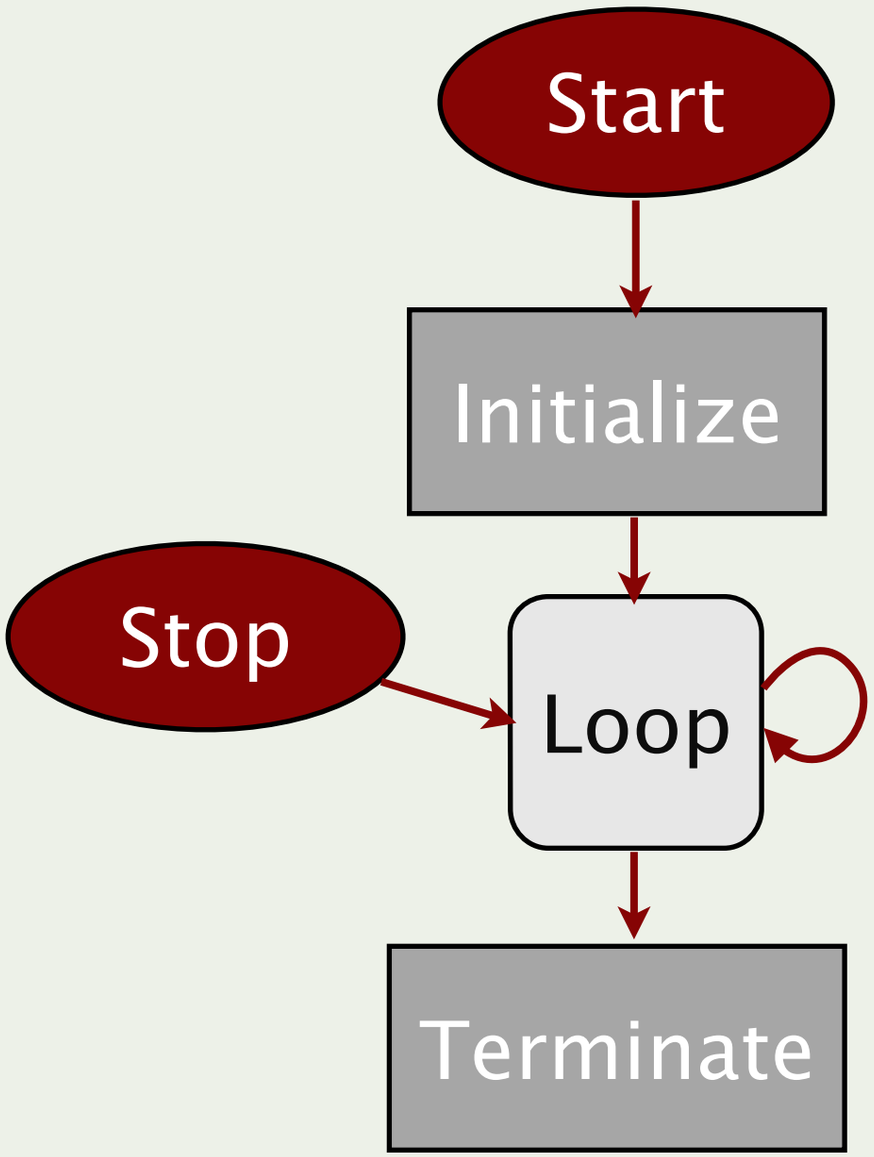



```
Pid2 ! {self(), Msg}
```

```
receive
  {From, start} -> …
  {From, stop}  -> …
end
```

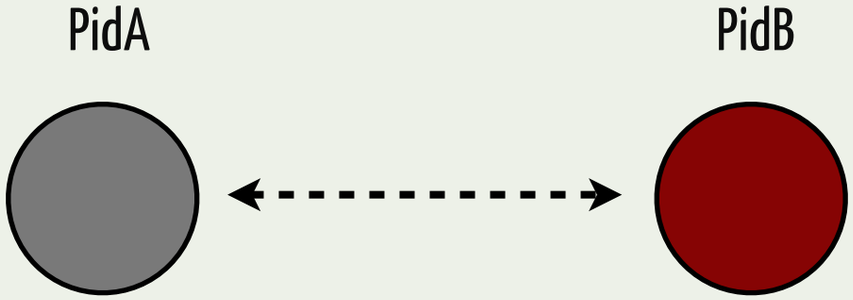
Receive

THE PROCESS SKELETON



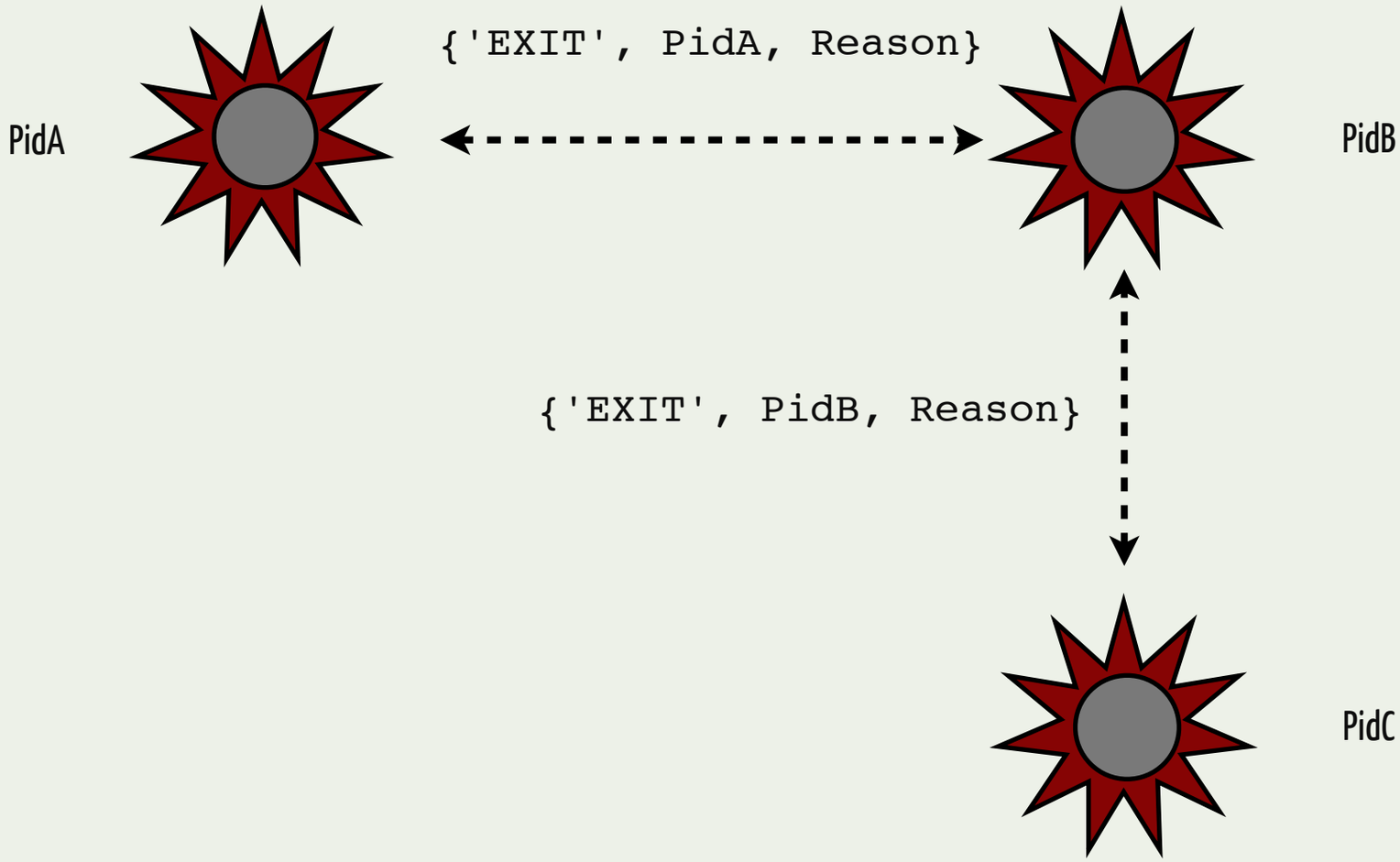

```
start(Args) ->
  spawn(server, init, [Args])

init(Args) ->
  State = do_init(Args),
  loop(State).

loop(State) ->
   receive
     {handle, Msg} ->
       NewState = handle(Msg, State),
       loop(NewState);
     stop ->
       terminate(State)
   end.

terminate(State) ->
  clean_up(State).
```

LINKS

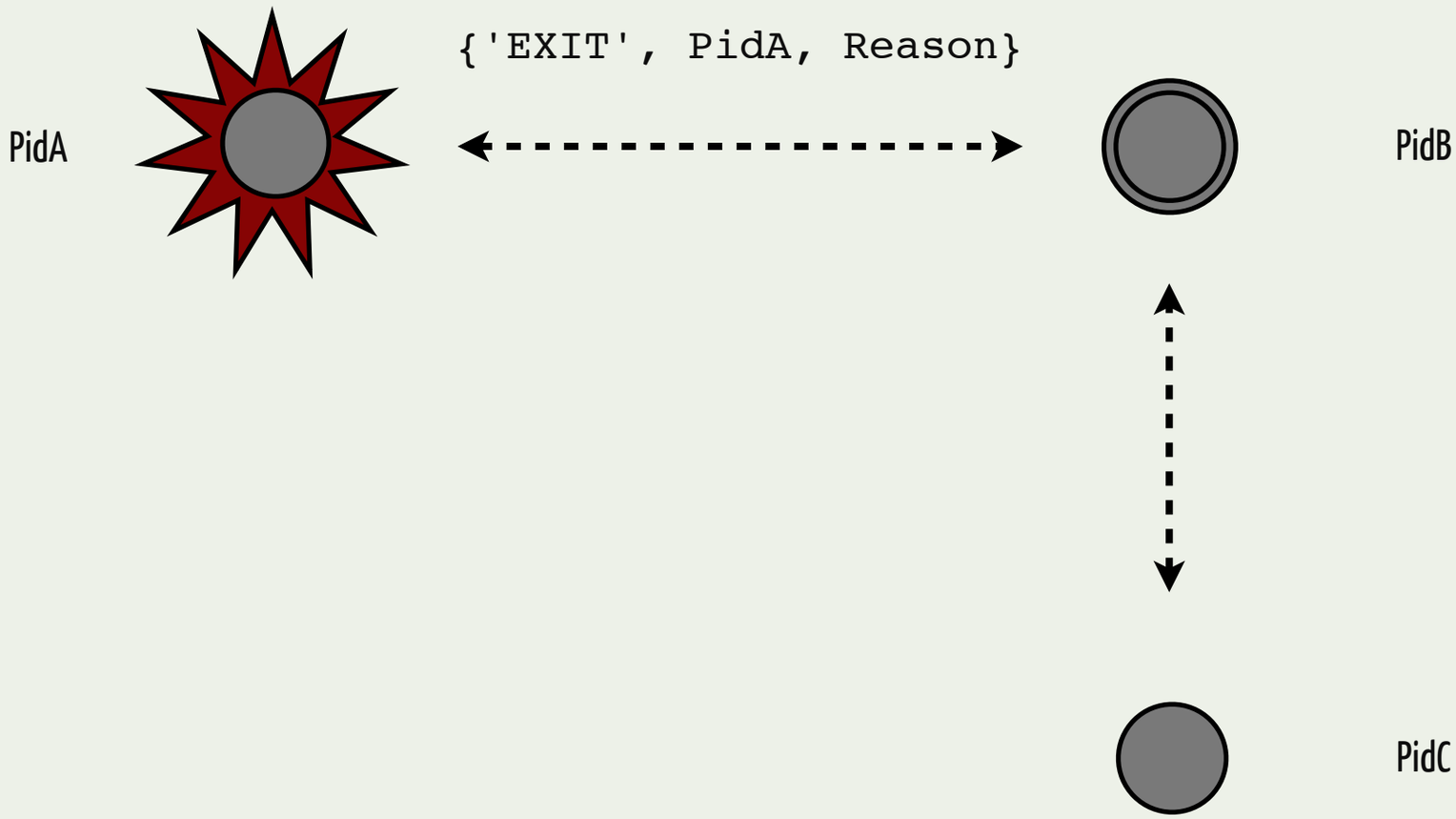


`link(PidB)`

EXIT SIGNALS

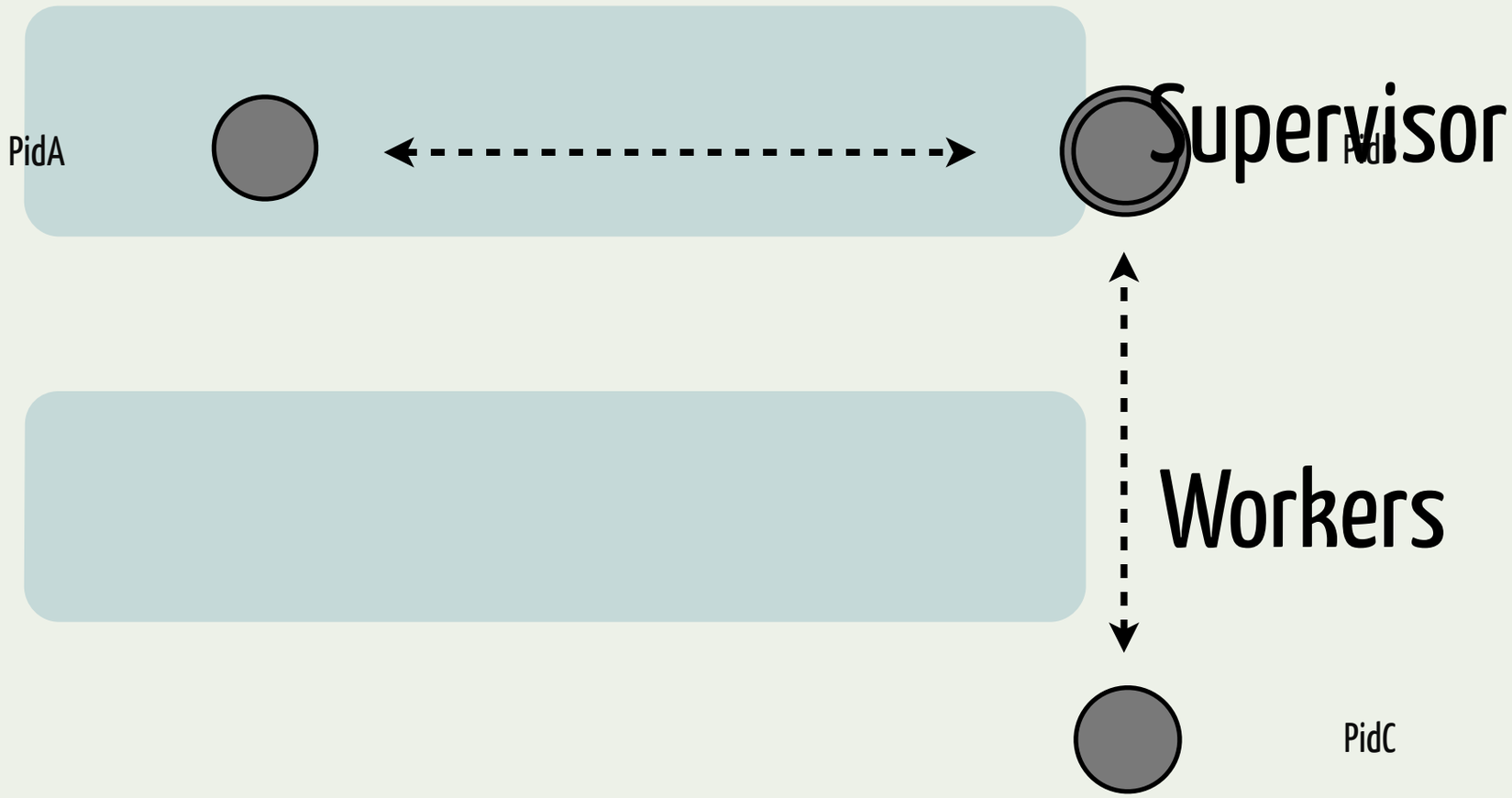

{'EXIT', PidA, Reason}
PidA
PidB
{'EXIT', PidB, Reason}
PidC

TRAPPING EXIT SIGNALS



process_flag(trap_exit, true)

SUPERVISORS


PidA
Supervisor
Pid
Workers
PidC

Where to Start

erlang.org	Official Home Page
github.com/erlang/otp	Sources
erlang-solutions.com	Binary Packages, News, Events
www.learnyousomeerlang.org	Best Online Tutorial
Erlang Programming	Best book about basics
Erlang and OTP in Action	Best book about OTP

Questions?

@robertoaloi

Cover Image: “How computers are made” (Reddit - Ness4114)