ERLANG / OTP

Paul Valckenaers

Processes and Messages

- spawn => een process aanmaken (= lightweight thread)
- ! => een boodschap sturen
- self() => de Pid (process-identifier) van het uitvoerend process
- receive => boodschappen uit de postbus halen en verwerken

Wat is een proces in Erlang?

- Een apart programma in uitvoering
  - … dat beschikt over een eigen (geheugen)ruimte.
  - … met de andere processen de beschikbare CPU(s) deelt.

Wat is een proces in Erlang?

- Een apart programma in uitvoering
  - … dat beschikt over een eigen (geheugen)ruimte.
  - … met de andere processen de beschikbare CPU(s) deelt.
- Een programma dat een Erlang functie uitvoert
  - … en dat afloopt als/wanneer die functie eindigt

spawn – maakt een proces aan

- spawn(Mod, Fun, Args)

- Mod is de module,
  Fun is een functie die Mod exporteert,
  Args is een lijst met de parameters die
  aan Fun worden meegegeven bij de start.

- Voorbeeld:
  spawn(teller, loop, [123]).

spawn – maakt een proces aan

- spawn(Mod, Fun, Args)

- Mod is de module,
  Fun is een functie die Mod exporteert,
  Args is een lijst met de parameters die aan Fun worden meegegeven.

- Voorbeeld:
  spawn(teller, loop, [123]).

```
-module(teller).
-export([...]).
-export([loop/1]).

loop(N) -> ...
```

spawn – geeft de proces identifier terug

- De functie-oproep

  spawn(Mod, Fun, Args)

  geeft de identifier (Pid) terug
  van het proces dat werd aangemaakt

- Normaliter vangt de aanmaker dit op:

```
Teller_1 = teller:create(),
Teller_2 = spawn(teller, loop, [0]).
```

```
-module(teller).
-export([create/0, ...]).
-export([loop/1]).

create() ->
    spawn(?MODULE, loop, [0]).

loop(N) -> ...
```

N.B. ?MODULE is een macro-oproep;
de compiler vervangt ?MODULE door teller.

- Demo

! – stuurt een boodschap

- Pid ! Msg

  stuurt de boodschap Msg naar
  het process met identifier Pid …

- Dit plaatst Msg in de postbus
  van het Pid process.
- De zender blokkeert niet;
  de ontvanger beslist hoe/wanneer
  de postbus wordt geledigd.

! – stuurt een boodschap

- Pid ! Msg

  stuurt de boodschap Msg naar het process met identifier Pid …

- Dit plaatst Msg in de postbus van het Pid process.
- De zender blokkeert niet; de ontvanger beslist hoe/wanneer de postbus wordt geledigd.
- Voorbeeld:

  teller:incr(Teller_1).

```
-module(teller).
-export([create/0, incr/1, reset/1, ...]).
-export([loop/1]).

create() ->
    spawn(?MODULE, loop, [0]).

incr(Teller_Pid) ->
    Teller_Pid ! incr.

reset(Teller_Pid) ->
    Teller_Pid ! reset.

loop(N) -> ...
```

N.B. in java-stijl: Teller_1.incr()

self() – Wat is de Pid van een proces?

- self() geeft de Pid van het proces dat die functie-oproep uitvoert.

- Ook in de Erlang shell:

```
self().
```

- Demo

self() – Wat is de Pid van een proces?

- self() geeft de Pid van het proces dat die functie-oproep uitvoert.

- Ook in de Erlang shell:

```
self().
1/0.
self().
self() ! yo.
flush().
```

receive – haalt boodschap uit postbus

- receive
  Msg -> … verwerk Msg …
  end

  haalt Msg uit de postbus en verwerkt de boodschap.

```
-module(teller).
-export([create/0, reset/1, incr/1, get/1]).
…
reset(TellerPid) -> TellerPid ! reset.
incr(TellerPid) -> TellerPid ! incr.
…
loop(N) ->
    receive
        reset ->   loop(0);
        incr ->    loop(N+1);
        …
    end.
```

- Demo

receive – haalt boodschap uit postbus

```
-module(teller).
-export([create/0, reset/1, incr/1, get/1]).
-export([loop/1]). % only for create/0

reset(TellerPid) -> TellerPid ! reset.

incr(TellerPid) -> TellerPid ! incr.

get(TellerPid) ->
        TellerPid ! {get, self()},
        receive
                N -> N
        end.
```

```
loop(N) ->
        receive
                {get, Pid} ->
                                Pid ! N,
                                loop(N);
                reset ->
                                loop(0);
                incr ->
                                loop(N+1)
        end.
```

Samenvatting – processen in Erlang

- spawn >> maak een nieuw proces aan
- ! >> stuur een boodschap
- self() >> geeft de Pid van uitvoerend proces
- receive >> verwerk een boodschap

Samenvatting – processen in Erlang

- spawn >> maak een nieuw proces aan
- ! >> stuur een boodschap
- self() >> geeft de Pid van uitvoerend proces
- receive >> verwerk een boodschap

1. Hoe laten we een teller-proces verdwijnen?
2. make_ref() geeft een unieke referentie terug.
   Waarvoor zouden we dat kunnen gebruiken?

Verbinding maken

- Dienstverlener_Pid ! {self(), Msg}

- loop(MyState) ->
  receive
  {Pid, Msg} ->
  Pid ! Antwoord(Msg),
  NewState = …
  loop(NewState);
  …
  end.

Verbinding maken

- Dienstverlener_Pid = spawn(M,F,[]),
  register(dienst_123, Dienstverlener_Pid),
  dienst_123 ! {self(), Msg}.

- register(dienst_abc, spawn(M,F,[]).

Verbinding maken

- Boodschap naar een Pid die niet meer bestaat

  NOP

- Boodschap naar een geregistreerde naam die niet (meer) bestaat geeft een fout (zender-crash)

- Registreer dus yellow and white pages i.p.v. …

Spawn_link

- Ontvanger bepaalt leegmaken van de mailbox
- After
- After zero, flush the mailbox.