Wissenswertes über Erlang

Guido Günther

2016-05-09

Who am I

- Free Software Hacker (Freelancing Software Developer)
- Debian Developer since 2000
- Contributed to libvirt-*, X11, Linux Kernel, GNOME, Calypso, ...
- FSFE Fellow and GNOME Foundation member
- Rather new to Erlang

Erl-what?

- Developed at Ericsson in 1986
- Open sourced 1998
- Ericsson Language or Agner Krarup Erlang?
- For telephone systems
- Functional, concurrent programming language
- Garbage collected runtime system
- Erlang/OTP

Erlang, Features

- Fault tolerant
- Hot code replacements
- Distributed
- highly available - 12 years of uptime - nine 9s (30 ms / a)

Projects

- ejabberd, Riak, CouchDB, RabbitMQ
- WhatsApp, GitHub, Facebook, . . .

You lack style - no, state!

Functional language

$f(x) = y$

$i = i + 3$

~~i = i + 3~~

```
for (int i=0; i < 3; i++) do_something(i);
```

~~for (int i=0; i < 3; i++) do_something(i);~~

```
1> A=3.
3
2> A=4.
** exception error: no match of right hand side value 4
```

No means No

- No imperative programming
- No global variables and state
- No pointers
- No variable reassignments

- Yes, Higher Order Functions
- Yes, Pattern Matching
- Yes, Code Reloads
- Yes, Lots of independent Processes

Recursion to the Rescue

```
loop(X) -> loop(X, 0).

loop(X=0, Sum) -> Sum;
loop(X, Sum)   -> loop(X-1, Sum+X).
```

Recursion in Detail

```
loop(3)   -> loop(3, 0)
loop(3,0) -> loop(3-1, 0+3)
loop(2,3) -> loop(2-1, 3+2)
loop(1,5) -> loop(1-1, 5+1)
loop(0,6) -> 6

3 + 2 + 1 = 6
```

Head to ~~Wall~~ Tail

```
A = [this, is, a, list].
[H|T] = A.
```

```
rev(L) -> rev(L, []).

rev([],    N) -> N;
rev([H|T], N) -> rev(T, [H] ++ N).
rev([a,b,c,d])        -> rev([a,b,c,d],      [])

rev([a|[b,c,d]], []) -> rev([b,c,d], [a] ++ [])

rev([b|[c,d]],  [a]) -> rev([c,d],  [b] ++ [a])

rev([c|[d]],  [b,a]) -> rev([d],  [c] ++ [b,a])

rev([d|[],  [c,b,a]) -> rev([], [d] ++ [c,b,a])

rev([],   [d,c,b,a]) -> [d,c,b,a]
```

More pattern matching

```
A = {this, is, a, tuple}.
{_, _, _, B} = A.
```

```
> B.
tuple
> _.
* 1: variable '_' is unbound
```

Fun with Funs

```
F = fun (X) -> X*X end.
> F(3).
> F(ok).
> F(3).
9
> F(ok).
** exception error: an error occurred when evaluating an ar
     in operator  */2
       called as ok * ok
```

Success, Typing!

Dialyzer

```
-spec foobar(atom()) -> [atom()].
foobar(App) -> ...
```

Processes, which Processes

Spawning a process

```
F = fun(X) -> io:format("~p~n", [X]) end.
times(X, Args, C) -> ...

spawn(l, times, [F,  "I'm a process", 3]).
spawn(l, times, [F,  "I'm a process", 3]),
spawn(l, times, [F, "I'm a process too", 3]).
```

Concurrency for free!

```
"I'm a process"
"I'm a process too"
"I'm a process"
"I'm a process too"
<0.53.0>
"I'm a process"
"I'm a process too"
```

Share(d) nothing

```
start_pong(Pid) -> ...
pong(Pid) -> ...

> P = l:start_pong(self()).
> P ! { ping, "foo" }.
> P ! giveup.
> R = fun () -> receive M -> M end end.
> R().
{pong,"hallo1"}
> R().
{pong,"hallo2"}
```

Let it crash

```
2> P ! {ping, 3}.
{ping,3}
3>
=ERROR REPORT==== 8-May-2016::15:13:30 ===
Error in process <0.34.0> with exit value:
{badarg,[{io,format,[<0.25.0>,"pong: Received ping '~s'~n"
         {l,pong,1,[{file,"l.erl"},{line,57}]}]}]}
```

```
2> P ! whatever
Unknwown message asfasdf, giving up
```

Linking processes

```
> process_flag(trap_exit, true).
> P = l:start_pong2(self()).
3> P ! sadfsadf.
Unknwown message sadfsadf, giving upsadfsadf
4> receive X -> X end.
{'EXIT',<0.35.0>,reason}
```

Distributed nodes

```
$ erl -sname ping
$ erl -sname pong
```

```
register(spawn(...))
```

On pong

```
> P = l:start_pong3().
```

On ping

```
> {pong, pong@bogon} ! {ping, "hello", self()}.
> {pong, pong@bogon} ! {ping, "hello", self()}.
> receive X -> X end.
{pong,"hello"}
```

```
erlang:node().
erlang:nodes().
```

Hot code replacements

```
version() -> ...
c(l).
```

OTP

Open Telecommunication Platform

Batteries inclued

- EUnit
- Mnesia
- HTTP/SNMP/. . .
- . . .

Behaviours

- gen_{server,fsm,event}
- supervisor
- application

Debugging

Tracing Processes, Messges, . . .

```
erl -sname observer -hidden  -run observer
```

Tracing functions

```
dbg:start().
dbg:tracer().
% trace messages (m)
dbg:p(pong, m).
```

The greater Erlang Universe

- REST: Webmachine
- JSON: jiffy
- YAML: p1_yaml
- Build, run, test: Rebar3
- . . . your favorite erlang app goes here . . .

Extending Erlang

NIFs - Gesundheit!

Native Implemented Functions

C-Nodes

See also

- Elixir
- LFE(Lisp flavored Erlang)
- Erlang on Xen

Further Reading

- Learn you some erlang - buy the book!
- On Debian /usr/share/doc/erlang-doc
- Everywhere else: https://www.erlang.org/docs

Questions?