

Erlang

Message Passing
Concurrency, For The Win

Toby DiPasquale

Commerce360, Inc.

Presented to Philly Linux Users Group

August 20, 2007

Er-what?

- Language/runtime created at Ericsson
- Designed for scalable, long-lived systems
- **Not** object oriented (depending on who you talk to)

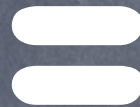
Another f***** language?

- Pattern matching
- Tail call optimization
- Message-passing concurrency
- Distributed programming
- Hot code update

Runway Models

- Meebo
- SlideAware
- RabbitMQ
- Jabber.org
- OpenPoker

Sequential



Not what you think it is, Performs a “binding”
Invokes pattern matching, Pattern matching like Prolog, nevermind Perl regexs

Syntax

- Variables start with uppercase letter
- Variables can only be assigned once
- Last evaluation is return value of function

She's got the look

```
-module(math_o_matics).  
-export([square/1]).
```

```
square(X) ->  
    X * X.
```


Atoms

- Self-indicating identifiers
- Start with lowercase letter
- Can also be quoted with single quotes

```
atom  
this_is_an_atom  
'I am also an atom'
```


Tuples

- Fixed length containers
- Often prepended with an identifier atom
- Decompose with pattern matching

```
Car = {car,  
      {honda, civic},  
      {horsepower, 100}}.
```

```
{car, Type, Power} = Car.
```


Lists

- Variable length containers
- Use `[H|T]` syntax to get head and tail of list

```
List = [1, 2, 3, four, 5.0]
```

```
[Head|Tail] = List
```

```
[H1,H2|T2] = List
```


Strings

- Sort of like in C
- Strings are just lists of integers
- Must use double quotes

```
Meeting = "PLUG".
```

```
Meeting2 = [80,76,85,71].
```


Arity

- Use functions with same name and different arity* as auxiliary functions

```
-module(math_o_matics).  
-export([sum/1]).
```

```
sum(L)           -> sum(L, 0).  
sum([], N)       -> N;  
sum([H|T], N)    -> sum(T, H+N).
```

* 'Arity' refers to the number of input parameters a function takes

Modules

- Logically associated code block
- Use colon (:) to use intermodule code
- Use `-import` to avoid prefixing

```
io:format("Using the module io~n").
```


The “fun” in functional

- Anonymous functions
- Used for higher-order programming

```
Square = fun (X) -> X * X end.
```

```
Cube = fun (X) -> Square (X) * X end.
```


List Comprehensions

- Takes an expression and a set of qualifiers and returns another list (like Python's)
- Looks like: $[X \mid Q_1, Q_2, \dots, Q_n]$

```
qsort([]) -> [];
```

```
qsort([Pivot|T]) ->
```

```
  qsort([X | X <- T, X < Pivot])
```

```
  ++ [Pivot] ++
```

```
  qsort([X | X <- T, X >= Pivot]).
```


Guards

- Simple tests against a pattern matching
- Makes code more concise and readable

```
max (X, Y) when X > Y -> X;  
max (X, Y) -> Y.
```

Multiple guards separated by a semicolon; Other guards include: arithmetic exprs, short-circuit boolean exprs, constants (False), true (True)

Biting the bits

- Syntax for extracting/packing bits
- Very handy for binary protocols (IPv4, MPEG, etc)

```
<<?IP_VERSION:4,  
    HLen:4, Srvctype:8, TotLen:16,  
    ID:16, Flgs:3, FragOff:13,  
    TTL:8, Proto:8, HdrChkSum:16,  
    SrcIP:32, DestIP:32, RestDgram/binary>>
```


Concurrency

Shared Memory



Image credit: <http://www.ja.org/nested/berrienandcass/kelly-vault.jpg>

Lots of contention in there; Sweaty and noisy after a while
Unpleasant; Teacher must watch to make sure things go smoothly (single point of failure)

Message Passing



Image credit: <http://english.people.com.cn/200512/21/images/pop2.jpg>

Pass messages to communicate; Stability through replication and communication
More like real world concurrent systems; No mutexes, semaphores, monitors, etc to deal with

Processes

- Basic unit of concurrency
- Managed by runtime, not OS (i.e. cooperative)
- Use `spawn/0`, `!/1` (a.k.a. `send`) and `receive/1` BIF's*
- Asynchronous send, synchronous receive

* BIF means "Built-in Function"

Owned by runtime, not OS; Very cheap and fast to spawn; No shared memory between processes
This is where some people would say Erlang is, in fact, OO: processes == objects (original sense of "object")

Concurrency Template

```
-module(template).  
-compile(export_all).  
  
start() ->  
    spawn(fun() -> loop([]) end).  
  
rpc(Pid, Query) ->  
    Pid ! {self(), Query},  
    receive  
        {Pid, Reply} ->  
            Reply  
    end.  
  
loop(X) ->  
    receive  
        Any ->  
            io:format("Received:~p~n", [Any]),  
            loop(X)  
    end.
```

Courtesy of Joe Armstrong in Programming Erlang, First Edition

loop/1 is tail-recursive
when you get a new message, add a clause to receive in rpc/2

Errors

- Linking processes defines error chain
- When a process dies, linked processes are sent an exit signal
- Use `spawn_link/1` to spawn linked processes

Distributing Erlang

- Erlang has built-in support for distributed operation
- Two modes:
 - Distributed Erlang (easier, less secure)
 - Socket-based distribution (more secure)

Distributing Erlang (2)

- Two libraries for higher-level Distributed Erlang:
 - rpc – RPC services
 - global – naming, locking, maintenance
- Cookie based security model

each machine is given a cookie; cookie must be same for all nodes in cluster
security is not great; suitable only for protected LANs

ets and dets

- Erlang Term Storage
- Dictionary for mad loads of Erlang data
- ets tables are RAM-based (transient)
- dets (disk ets) tables are persisted to disk

support insertion/lookup of keys and deletion of entire table

Mnesia

- Real-time, distributed database that comes with Erlang
- Query language looks like a lot like SQL/list comprehensions
- Built-in visualization tools

OTP

- Open Telecom Platform
- Web server, FTP server, CORBA ORB, ASN.1, SNMP, etc
- Designed around encapsulated “behaviors”

Has lots more than telco stuff; set of libraries and platforms for building large-scale apps
behaviors help code up large-scale apps (e.g. `gen_tcp`, `gen_server`)

Hot Process-on-Process action

- Yaws

- Super scalable Web server/platform

- ejabberd

- Super scalable XMPP (Jabber) server

- RabbitMQ

- Super scalable message broker

RTFM

- Programming Erlang (PDF and dead tree versions; great book)
- Concurrent Programming with Erlang (older; first half available online at no cost)
- Erlang Website
- Trapexit forums
- erlang-questions mailing list

Huh huh huh huh... you said
'Erlang'

Slides: <http://cbcg.net/talks/erlang.pdf>

Me: toby@cbcg.net