DIT027: Distributed Fault-tolerant programming

2012

Erlang – Installation and Introduction

Hans Svensson
hanssv@ituniv.se

Topics

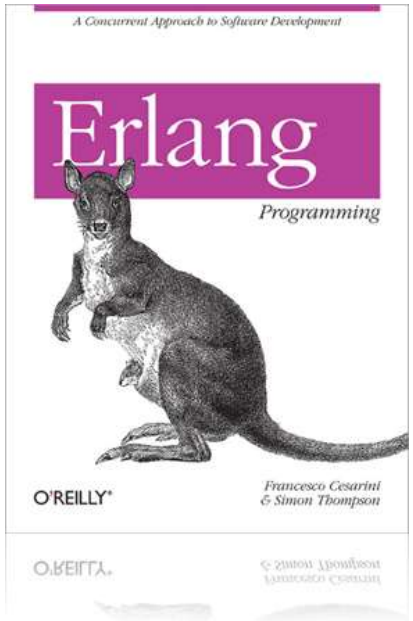
- Erlang Course (DIT027) information
- Install (Erlang and) Emacs
- Edit – Compile – Run Erlang code
- Basic Erlang – Examples

DIT027 – Course Information

- **Teachers**
  - Hans Svensson (hanssv@ituniv.se)
    - Course responsible
  - Francesco Cesarini (francesco@erlang-solutions.com)
    - Lecturer

O'Reilly currently has a "Back to School" discount (40 or 50%). Use discount code: B2S2

- **Literature**
  - ***Erlang Programming*** *(Cesarini & Thompson)*
  - http://www.erlangprogramming.org/



DIT027 – Course Information

- **Workshops**
  - Erlang introduction – 5/9 9:00 – 12:00
  - Sequential Erlang – 10-11/9 9:00 – 17:00
  - Repetition Seq. Erlang – 26/9 9:00 – 12:00
  - Distributed Erlang – 3-5/10 9:00 – 17:00
  - Repetition Dist. Erlang – 17/10 9:00 – 12:00

- **Exam**
  - Erlang exam – 24-26/10 (exact date not confirmed)

DIT027 – Course Information

- **Examination**
  - Written exam. Earlier exams provides some idea about what to expect.
    - Max 100 credits
    - Pass /Godkänt (G) >= 50%
    - Pass w. distinction/Väl godkänt (VG) >= 75%
  - Submission of correct and original solutions to assignments gives extra credits for the regular exam (not valid for later re-examinations). Maximal three (3) credits for each assignment.

DIT027 – Course Information

- **Optional Assignments (individual)**
  - Assignment 1 – Sequential Erlang
    - **Deadline – 30/9 23:59**
  - Assignment 2 – Distributed Erlang
    - **Deadline – 21/10 23:59**
- **Q&A sessions**
  - **4 sessions, tentatively scheduled for:**

    20/9, 27/9, 11/10, and 18/10. Between 13 and 15
  - Supervisors will be available for your questions – go there if you've got problem with Erlang.

Reminder: Deadline

dead·line [ded-*lahyn*] – *noun*

1. the time by which something must be finished or submitted; the latest time for finishing something: *a five o'clock deadline*.
2. a line or limit that must not be passed.
3. a boundary around a military prison beyond which a prisoner could not venture without risk of being shot by the guards.

"The submission system seems to be broken, I attach my solutions in this mail"

- Sent by a student at 8:25 (deadline 8:00)

DIT027 – Course Information

- **Organizational changes**
  - I will not be around on a day-to-day basis.
  - Therefore, we will use GUL for questions rather than E-mail and personal contact. Supervisors assist with answering questions
  - Apart from this, most things will be just like last year when the course was well received (3.82 in average rating!)

DIT027 – Course Information

- **Course evaluation – last year's comments**
  - Good to have repetition workshops in between Francesco's visits.
  - Assignments were quite hard, but helped the learning a lot.
  - Very, very, intensive days with Francesco, but good preparation for the future.
  - The written exam was well received.

Install Erlang

Hopefully you've already done this…

- http://www.erlang.org/download.html
  - Current stable version is R15B01
- Windows:
  - Precompiled binaries in a .exe file
  - Just run the installer and you are good to go!

Install Erlang

- Linux:
  - Most distributions include Erlang as a package
    - Often an older version (not a problem today, but maybe later in the project…)
    - Ubuntu: currently at R14B04 and has 45+ Erlang packages
  - Alternative: download binaries or compile yourself
    - For configuration and details, Google is your friend!

```
wget http://erlang.org/download/otp_src_R15B01.tar.gz
tar zxvf otp_src_R15B01.tar.gz
cd otp_src_R15B01
./configure && make && sudo make install
```

Install Erlang

- Linux:
  - Another (the best?) alternative: Use packages from Erlang solutions:
    - https://www.erlang-solutions.com/downloads/download-erlang-otp

Install Erlang

- Mac OS/X:
  - From source:
    - See for example: http://wiki.basho.com/Installing-Erlang.html for help
    - For configuration and details, Google is your friend!
  - Alternative: use Homebrew or download binaries
    - See for example: http://simergence.blogspot.com/2010/04/erlang-on-mac-for-lazy.html
  - Or Erlang Solutions:
    - https://www.erlang-solutions.com/downloads/download-erlang-otp

The Erlang shell

- Let's try our first bits of Erlang. Start the Erlang shell (a read-eval-print-loop).
  - Linux: `erl`
  - Windows: `werl.exe`
- Try some simple expressions
- Don't forget the almighty dot '.' at the end of expressions.

Simple arithmetics

```
Erlang R14B03 (erts-5.8.4) [smp:4:4] [rq:2] [async-
threads:0]

Eshell V5.8.4  (abort with ^G)
1> 12 + 4.
16
2> 3 + 4 * 5.
23
3> (3 + 4) * 5.
35
4> 127 - 18
4> .
109
```

Large numbers

```
…
17> 732888923947729 * 1718887283846 *
2758889215787349992045653.
34755202128469684538822486481336861273518800 1
2014302
18> 16#cafe * 32#sugar.
1577682511434
19>
```

Using base 16
and base 32!!

Variables – a little surprise

Variable name starts with capital letter!

```
…
3> X = 1234567890.
1234567890
4> X.
1234567890
5> X * X * X + X.
1881676371789154862131636890
6> X = 1234.
** exception error: no match of right hand
side value 1234
7>
```

What!?

Variables – the shocking truth

- What is going on here, why couldn't we assign a value to our variable X ?

- X is not a variable (at least not in the way variables are used in for example Java).
- = is not the assignment operator

- Francesco will explain this in more detail on Monday

Editor

- **Emacs** (recommended)
  - Lightweight, very customizable
  - Erlang mode included in Erlang distribution
  - Quite a steep learning curve…
- **Eclipse** (also good, especially later)
  - Don’t miss the Erlang-mode (Erlide)
  - A little large and slow for netbooks

Editor

- **vi/vim** (recommended)
  - Lightweight, very customizable
  - Erlang mode exists
  - Very steep learning curve…
- **gEdit/notepad++/kate/Textmate**
  - Simple editors with syntax highlighting
- **Your favorite editor…**

Editor – a side note

- When submitting assignments (and the project) – make sure not to depend on the editor
- Development environments often hide details, which is great **when you know the details!**
- We strongly recommend to use a simple editor first, to understand how compilation, etc works. (gedit, vim, Emacs, …)

Emacs*

- Linux: Included (or addable) in all distributions
- MacOS: Aquamacs is essentially Emacs.

  http://aquamacs.org/
- Windows: Binary distribution (zip-file)

  ftp://ftp.sunet.se/pub/gnu/emacs/windows/
  - Current version: emacs-24.1-bin-i386.zip

  Warning: Don't use ALZip for unzipping, it is broken!

  Warning2: Don't select the barebin zip-files!

* *"a great operating system, lacking only a decent editor"*

Emacs – Configuration Linux/MacOS

- Create a file `.emacs` in you home directory
- Add the following lines to your `.emacs`

```
(setq load-path
  (cons "/path/to/erlang/lib/tools-<ToolsVer>/emacs"
        load-path))
(setq erlang-root-dir "/path/to/erlang")
(setq exec-path (cons "/path/to/erlang/bin"
                      exec-path))
(require 'erlang-start)
```

- Further instructions:

  http://www.erlang.org/doc/apps/tools/erlang_mode_chapter.html

Emacs – Configuration Windows

DISCLAIMER: Only tested for Windows 7

- Unzip the files
  - As an example in D:\Program\Emacs
- Add an environment variable: HOME
  - For instructions see http://www.itechtalk.com/thread3595.html
  - HOME should be set to wherever your home directory is located (mine is C:\Users\Hans)
- If you are running Windows XP - Reboot

Emacs – Configuration Windows

- In your HOME-directory, create a directory named: **.emacs.d**
- Create a file **init.el** in the new directory

  I.e. C:\Users\Hans\.emacs.d\init.el
- Add the following to your **init.el**

NOTE: Forward slashes!!
Adapt to where you installed Erlang!

```
(setq load-path
  (cons "D:/Program/Erlang/erl5.9.1/lib/tools-2.6.7/emacs"
        load-path))
(setq erlang-root-dir "D:/Program/Erlang/erl5.9.1")
(setq exec-path
   (cons "D:/Program/Erlang/erl5.9.1/bin" exec-path))
(require 'erlang-start)
```

Emacs – Configuration Windows

- Associate **.erl** files with Emacs
  - Right-click on a **.erl**-file, then select 'Open with', and browse for **runemacs.exe** remember to check the box 'Always use this program for …'
- Test your configuration with one of the example files (available in GUL). Once open in Emacs the code should be nicely colored and there should be an 'Erlang' menu at the top!

- Further instructions:
  http://www.erlang.org/doc/apps/tools/erlang_mode_chapter.html
  or use Google!

Edit Erlang Code – in Emacs

- Create a file: C-x C-f (Means first press Ctrl+x then Ctrl+f) – or use the menu.
- Save file: C-x C-s (or use the menu)
- Search in buffer (file): C-s
- Undo: C-_ (that is an underscore!)

- Emacs should indent your code automatically
- Skeletons, etc are available in the 'Erlang'-menu
- You won't master Emacs in one hour/day/year…

Code examples

- All code examples mentioned are available under 'Documents' and 'Erlang Introduction' at the course page in GUL

Compile Erlang code

- Linux
  - Command line: `erlc filename`
  - or use the Erlang shell (next slide…)
  - or compile from Emacs (select in menu)
- Windows
  - Use the Erlang shell (next slide…)
  - or compile from Emacs (select in menu)

Compile Erlang code

- Start the Erlang shell, compile **test_ex1.erl**:

```
…
1> c(test_ex1).
{ok,test_ex1}
2>
```

c is short for compile!

- You must be in the right directory for this to work…

```
…
3> pwd().
<your-current-directory>
4> cd("C:/Path/to/where/the/code/is”).
```

NOTE: Forward slashes!!

Handy trick for Windows users

- If you just start Erlang (**werl.exe**) from the Start-menu, you end up in the directory where you installed Erlang… Not very good!
- Trick: Associate **.beam**-files (Compiled Erlang code is placed in **.beam**-files) with **werl.exe**
- Now if you want to start a shell in a directory where there is already compiled code (or simply create an empty .beam-file in the directory, the shell won't care) and double-click it. Voila; you get a shell where the current directory is set!

Practice makes perfect

- While learning how to write Erlang code, you'll make mistakes (lots of them!)
- A 'compile – run – see error – make code change – recompile – rerun'-cycle should not take time in the (re-)compile and (re-)run phases!!

- Horror stories from the past…

Our first Erlang module

```
%%% File    : test_ex1.erl
%%% Author  :  <Hans@HANS-LAPTOP>
%%% Description : Hello World
%%% Created :  8 Sep 2009 by  <Hans@HANS-LAPTOP>
-module(test_ex1).

-export([hello/0]).

%% Hello world -- in Erlang.
hello() ->
    io:format("Hello world!\n").
```

Run Erlang code (test_ex1)

- In the Erlang shell

```
…
12> test_ex1:hello().
Hello world!
ok
13>
```

Module name (=filename)

Function name

Program output (io:format(…))

Function return value

Erlang basics – datatypes

- Datatypes
  - Integers (1, 14, -254, 2#0101, 16#4CF5)
  - Floats (3.14, -2.91, 2.34e5)
  - Atoms (foo, undefined, true, false, whatnot)
  - Variables (Foo, X, Y, LongVariableName)
  - Tuples ({1,2}, {A,B,{C,D}}, {person,Name,Age})
  - Lists ([], [1,2,3], [foo,X,3,false], [[1],[2,3]])
  - Process identifiers (<0.12.145>, <123.234.12>)

Erlang basics – datatypes

- Strings are just lists of integers
  - `"Hello" => [72,101,108,108,111]`
- Variables are single assignment only
  - `X = 12.`
- Records are just special tuples
  - `{person,"Kalle",23}`
- Lists consists of a **head** and a **tail**.
  - `[Head | Tail]`
  - `[1,2,3,4] = [1 | [2 | [3 | [4 | []]]]]`

Erlang basics – Pattern matching

- '=' means pattern matching
- Pattern matching is common in functional languages
- <Left-hand-Side> = <Right-hand-Side>
  - Succedes if Lhs and Rhs matches
  - Free variables in Lhs that matches are bound
  - Example: {X,Y} = {12,24}. Matches if X and Y are previously unbound (or X = 12 and/or Y = 24). Afterwards X = 12 and Y = 24.

Erlang basics – Pattern matching

In the following sequence, what matches succeed?

```
true = true.
true = false.
X = 12.
Y = hello.
X = 18.
{true,Y} = {true,hello}.
{X,Y} = {hello,18}.
{A,B} = {333,666,999}.
{A,B,A} = {333,666,999}.
{A,B,A} = {333,666,333}.
[H | T] = [1,2,3].
[H2 | T2] = [15].
[H3 | T3] = [].
[H4 | T4] = ”Hello”.
```

Erlang basics – Pattern matching

In the following sequence, what matches succeed?

```
true = true.                    OK
true = false.                   Fails
X = 12.                         OK, X = 12
Y = hello.                      OK, Y = hello
X = 18.                         Fails, 12 /= 18
{true,Y} = {true,hello}.        OK
{X,Y} = {hello,18}.             Fails, 12 /= hello
{A,B} = {333,666,999}.          Fails, different structure
{A,B,A} = {333,666,999}.        Fails, 333 /= 999
{A,B,A} = {333,666,333}.        OK, A = 333, B = 666
[H | T] = [1,2,3].              OK, H = 1, T = [2,3]
[H2 | T2] = [15].               OK, H2 = 15, T2 = []
[H3 | T3] = [].                 Fails
[H4 | T4] = "Hello".            OK, H4 = 72, T4 = [101,...]
```

A second example – Shopping*

```
%%% File    : shop.erl
%%% Author  :  <Hans@HANS-LAPTOP>
%%% Description : A little shop module, inspired
%%%               Joe Armstrong.
%%% Created :  6 Sep 2010 by  <Hans@HANS-LAPTOP>

-module(shop).

-export([cost/1]).

%% Cost of things
cost(apple)  -> 3;
cost(banana) -> 5;
cost(milk)   -> 2;
cost(orange) -> 8;
cost(soda)   -> 6.
```

*The example can be found in *Programming Erlang* by Joe Armstrong

A second example – Shopping

```
…
9> c(shop).
{ok,shop}
10> shop:cost(apple).
3
11> shop:cost(milk) + shop:cost(banana).
7
12> shop:cost(coconut).
** exception error: no function clause
matching shop:cost(coconut)
13>
```

- OK, great, but we need a little bit more

A second example – Shopping

```
…
-module(shop).

-export([cost/1,total/1]).

cost(apple) -> 5;
…

%% Total sum of purchase
%% A purchase is a list of tuples {<item>, <amount>}
total([{Item, N} | T]) ->
    cost(Item) * N + total(T);
%% The cost of no items is 0
total([]) ->
    0.
```

A second example – Shopping

```
…
13> c(shop).
{ok,shop}
14> shop:total([]).
0
15> shop:total([{apple,4}]).
12
16> shop:total([{milk,1},{apple,4}]).
14
17> shop:total([{coconut,5}]).
** exception error: no function clause matching
shop:cost(coconut)
     in function  shop:total/1
18> shop:total([{apple,2},{milk,1},{apple,4}]).
20
```

How total() works

- So how does that function total, manage to compute the sum of the purchase!?
- Let us evaluate:

```
shop:total([{milk,1},{apple,4}]).
```

In the call to **`total`**, we match

`[{Item, N}|T]` against **`[{milk,1},{apple,4}]`**

This means that:

`Item = milk, N = 1`, and **`T = [{apple,4}]`**

Next we compute

```
cost(Item) * N + total(T)
```

where:

`cost(milk) * 1 = 2 * 1 = 2` and

How total() works

…

and **`total([{apple,4}])`** means a new call to total, where we match

`[{Item, N} | T]` against **`[{apple,4}]`**

This means that:

`Item = apple`, **`N = 4`**, and **`T = []`**

Next we compute

`cost(Item) * N + total(T)`

where:

`cost(apple) * 4 = 3 * 4 = 12` and

`total(T) = total([]) = 0.`

Thus the end result is: **`2 + (12 + 0) = 14.`**

A more general principle

- This way of computing is a much more general principle, which you'll see a lot in this course.
- To process a list we use:

```
process([Head | Tail]) ->
   some_function(Head) + process(Tail);
process([]) ->
   0.
```

- Francesco will teach you all about this!

A small exercise

- The ‘broken’-module is broken! The compiler will give a warning. Fix the program!

```
-module(broken).

-export([add/2,sum/1]).

%% Function that adds two numbers
%% HINT: This function is seriously broken!
add(a, b) ->
    a + b.

%% Function that sums a list of numbers
%% HINT: There is a small mistake here
sum([H | T]) -> H + sum(T);
sum([]) -> 1.
```

A small exercise

```
…
29> c(broken).
./broken.erl:13: Warning: this expression will fail
with a 'badarith' exception
{ok,broken}
30>
```

- Note that the compiler only warns you of one error. There are two errors in the code.
- Once you have fixed both errors, re-name your module to 'fixed'. Make sure that you can still compile it after re-naming.

To end things

- Now you should be prepared for the first visit by Francesco!
- Come prepared on Monday:
  - Be on time
  - Be well rested
  - Bring your laptop

- By the way, to quit the Erlang shell, use: `q().` (or `halt().`)

A small exercise - solution

```
-module(fixed).

-export([add/2,sum/1]).

%% Function that adds two numbers
add(A, B) ->
    A + B.

%% Function that sums a list of numbers
sum([H | T]) -> H + sum(T);
sum([]) -> 0.
```