CS 683 Emerging Technologies
Fall Semester, 2008
Doc 1 Introduction & Erlang
Sept 2 2008

Copyright ©, All rights reserved. 2008 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/openpub/) license defines the copyright on this document.

References

Programming Erlang: Software for a Concurrent World, Armstrong, Chapter 2.

History of Erlang, Armstrong, http://delivery.acm.org/10.1145/1240000/1238850/supp/Erlang.pdf?key1=1238850&key2=8900130221&coll=GUIDE&dl=GUIDE&CFID=61517297&CFTOKEN=84428823

Reading

Programming Erlang: Software for a Concurrent World, Armstrong,
Chapter 2 & 3 for Thursday Sept. 4

2

Introduction

3

Course Issues

http://www.eli.sdsu.edu/courses/index.html

Crashing
Course Web Site
Wiki
Screen Casts
Prerequisites
Grading
First year student warning
Cloud Computing Accounts
Rough edges

4

Topics

Erlang

Cloud Computing

Google's Android - Mobile Computing

5

Frustration

6

The Web is 6317 Days Old

May 17, 1991

first release of WWW code on CERN machines

What will the web be like in another 6000 days?

7

Change

Incremental
versus
Revolutionary

8

Old Way

Programmed one processor

Main Frame
Minicomputer
PC
Mobile device

Processor may talk to other processor

9

Trends - Multicores

In the near future each Intel processor will have 1000s cores

10

Trends - Computing in the Cloud

Where are computers running Google Apps

Trends - iPhone

Third party developers can make money selling iPhone apps

12

Global Computer

Billions of processors
100 exabytes of storage

Our devices are part of the GC

http://www.ted.com/index.php/talks/kevin_kelly_on_the_next_5_000_days_of_the_web.html

13

Erlang

14

Download

http://www.erlang.org/

15

Why Erlang

Highly reliable

Highly scalable across multiple processors

Pay attention to Erlang's model of concurrency

16

History

Better way to program telephony

1982-86 Ericsson Labs

Experiments with programming languages

1988

First use of Erlang in Ericsson outside of labs

1993

Erlang becomes available outside of Ericsson

1997

Bit syntax & binaries add for protocol programming

1998 - Erlang become open source

2006 - SMP Erlang

17

Requirements

Very large number of concurrent activities

Real time

Systems distributed over several computers

Interaction with hardware

Very large software systems

Continuous operation over several years

Reconfiguration without stopping the system

Stringent quality and reliability requirements

Fault tolerance both to hardware failures and software errors

18

Erlang Features

Concurrent

Very light-weight concurrency

“Share nothing” process semantics

Pure asynchronous message passing

Dynamically typed functional programming language

Non-pure extensions for databases

Mechanisms for in-service code upgrade

Set of libraries (OTP)

19

Rohan

Tools

compiler
/usr/local/bin/erlc

erlang shell
/usr/local/bin/erl

Documentation

web
http://www-rohan.sdsu.edu/doc/R12B/doc/

Man pages
man erl
man erlc
man escript

20

Hello World

File: greeting.erl

```
-module(greetings).
-export([hello/0, print_hello/0]).

hello() ->
      "Hello World".

print_hello() ->
      io:format("~w ~n", [hello()]).
```


21

Erlang Shell

```
Al pro 44->erl
Erlang (BEAM) emulator version 5.6.3 [source] [smp:2] [async-threads:0]
[kernel-poll:false]

Running ErlangEshell V5.6.3  (abort with ^G)
1> c(greetings).
{ok,greetings}
2> greetings:hello().
"Hello World"
3> greetings:print_hello().
[72,101,108,108,111,32,87,111,114,108,100]
ok
4> halt().
Al pro 45->
```


22

In Unix Shell

```
Al pro 37->ls

greetings.erl

Al pro 38->erlc greetings.erl

Al pro 39->ls

greetings.beam   greetings.erl

Al pro 40->erl -noshell -s greetings hello -s init stop

Al pro 41->erl -noshell -s greetings print_hello -s init stop

[72,101,108,108,111,32,87,111,114,108,100]
```


23

Variables

```
1> X = 2 + 4.
6
2> Y = 12.
12
3> X = 10.
** exception error: no match of right hand side value 10
4> X = Y
4> ,
4> .
* 3: syntax error before: '.'
4> X = Y.
** exception error: no match of right hand side value 12
```

Start with Capital letter

Do not change

24

Atoms

hello
cat
'rat'
'AnAtom'

Named constant
No value
Start with lowercase or
is single quoted

25

Tuples

```
6> OffCenter = {point, 2, 5}.
7> Person = {person, {name, "roger"}, {office, gmcs561}}.
{person,{name,"roger"},{office,gmcs561}}
8> {point, Xaxis, Yaxix} = OffCenter.
{point,2,5}
9> Xaxis.
2
10> {person, {name, "roger"}, Location} = Person.
{person,{name,"roger"},{office,gmcs561}}
11> Location .
{office,gmcs561}
```

Anonymous struct
atom-value pairs convention

26

Lists

store variable numbers of things

[1, 2, 3, "cat"]
[{apples,10}, {oranges,3},{mangoes,1}]

13> [Head | Tail] = [1, 2, 3, 4].
14> Head.
1
15> Tail.
[2,3,4]

27

Bound & Unbound Variables

X = 5

Y = 10

List = [X, Y]

Error = [X, Z]

Bound variable

Has a value

Unbound variable

No assigned variable

28

Pattern Matching

```
1> {X, abc} = {123, abc}
1> .
{123,abc}
2> X.
123
3> [A, B, C |T] = [a, b, c, d, e, f].
[a,b,c,d,e,f]
4> A.
a
5> T.
[d,e,f]
```

=

pattern matches left and right side

Not assignment

29