Les bases d'Erlang

Programmation Fonctionnelle
Master 2 I2L apprentissage

Sébastien Verel
verel@lisic.univ-littoral.fr
http://www-lisic.univ-littoral.fr/~verel

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe OSMOSE

Septembre 2016

Plan

1. Calculatrice
2. Types composés
3. Patern matching et fonction
4. Structures conditionnelles

Avertissement

Ce premier cours n'est pas un cours à proprement parler un cours de programmation fonctionnelle
Introduction à la syntaxe d'un langage fonctionnel, le langage Erlang

Pourquoi Erlang ?

- Langage paradigme de **programmation fonctionnelle**
- Gestion des processus, concurrence (envoie de message), informatique distribuée, temps réel souple, architecture multicoeur :
  cf. cours semestre 2
- **Robustesse**

Historique

Historique

- **1980** laboratoire Ericsson, nouveau langage pour programmation de nouveau produit telecom
- Influence langages fonctionnels (ML, Miranda), langages concurrents (Ada, Modula), programmation logique Prolog
- Machine virtuelle écrite en prolog
- **1991** machine virtuelle en C
- Lancement en **1994**, amélioration en 1995 : réseaux large bande, GPRS, commutateur ATM
- Besoin "immédiat" systèmes distribués, tolérants aux pannes, massivement concurrent, temps réel souple
- **1998**, open-source, licence EPL dérivée licence publique Mozilla

Bibliographe / Webographie

- "Programmer en Erlang", Francesco Caserini, Simon Thompson, (trad. Eric Jacoboni), Pearson, 2010.
- Learn you some Erlang : `http://www.learnyousomeerlang.com/content`

Start / Stop

Lancement

Lancer le shell Erlang par :

```
erl
```

Expression

Tout ce termine par un point .

```
q().
```

Nombres

Entiers

Classique :

```
801.
-10.
```

base♯nombre.

```
2#1101.
16#FF.
16#A0.
```

Nombres

Flottants

17.5.
-5.0.
6.022E23.
3.02E-2.

Précision sur 64 bits.

Opérateurs arithmétiques binaires

Opérateur	description	types
$+$	addition	Entier, flottants
$-$	soutraction	Entier, flottants
$*$	multication	Entier, flottants
$/$	division à virgule flottante	Entier, flottants
div	division entière	Entiers
rem	reste de la div entière	Entiers

A vous !

Exercice

Calculer (le rapidement possible) les nombres suivants :

A vous !

Exercice

Calculer (le rapidement possible) les nombres suivants :

a $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10$

b le carré de l'approximation à 10^{-5} du nombre π

c Le nombre de secondes de l'année 2016

d Le nombre d'atomes contenus dans une mole d'eau

e Le temps (à la minute près) mis par un véhicule pour aller jusqu'à la lune en ligne droite depuis la Terre à la vitesse moyenne de 260 km/h.

f $22/7$

Atomes

Définition

Symboles littéraux qui ne représentent qu'eux-mêmes (cf. Prolog).
Même rôle que les types énumérés.

Synthaxe

- Commence par une minuscule
- ensuite lettres, chiffres, arobases, points, caractères soulignés
- Placé entre apostrophes : n'importe quel caractère.

Atomes

Définition

Symboles littéraux qui ne représentent qu'eux-mêmes (cf. Prolog).
Même rôle que les types énumérés.

Synthaxe

- Commence par une minuscule
- ensuite lettres, chiffres, arobases, points, caractères soulignés
- Placé entre apostrophes : n'importe quel caractère.

Tests immédiats

septembre.
aMoi.
super_bien_ca.
i_love_Calais.
'Ca Commence Par là'.

Booléens

Synthaxe

Pas de type spécial !

Seulement atomes `true` et `false` avec les opérateurs booléens

Opérateur	description
`not`	négation
`and`	et logique classique
`andalso`	renvoie false dès la premier opérande vaut false (sans évaluer le second)
`or`	ou logique
`orelse`	renvoie true dès la premier opérande vaut true (sans évaluer le second)
`xor`	or exclusif

Variables

Synthaxe

- Commence par une majuscule
- ensuite lettres, chiffres, arobases, points, caractères soulignés

Affection unique

Dans la porté d'une fonction (et donc processus du shell)
On ne peut modifier une variable en Erlang
(bizarre non pour une variable ?)

Variables

Synthaxe

- Commence par une majuscule
- ensuite lettres, chiffres, arobases, points, caractères soulignés

Affection unique

Dans la porté d'une fonction (et donc processus du shell)
On ne peut modifier une variable en Erlang
(bizarre non pour une variable ?)

Test immediat

$A = 1.$
$A = A + 1.$
$NewA = A + 1.$
$4 = 2 + 2.$
$3 = 2 + 2.$

Variables

Quelques remarques et précisions

- Tous les appels de variables se font par *valeur*
- Il n'existe pas d'appel par *référence*
- Les variables sont locales à la fonction dans laquelle elles sont liées (notion d'environnement)
- Pas de variables globales
- Pas de déclaration
- Typage dynamique

Test immediat

Dans le shell on peut utiliser la fonction $f()$ pour délier une variable :
A = 1.
f(A).
A = 2.

Comparaison de termes

Opérateurs

Opérateur	description
==	égale à
/=	différent de
=:=	exactement égale à
=/=	non exactement égale à
=<	inférieur ou égale à
<	strictement infére ieur à
>=	supérieur ou égale à
>	strictement supérieur à

Ordre entre types

nombre $<$ atome $<$ référence $<$ fun $<$ port $<$ pid $<$ tuple $<$ liste $<$ binaire

Test immediat

```
1 == 1.0.
1 /= 1.0.
1 =:= 1.0.
1 =/= 1.0.
```

Exercice

Exercice

Pour $x = 5$, $a = (x \geq 12)$, $b = (x \leq 2)$, $c = (x < 6)$.

- Quelle est la valeur des expressions suivantes :
  - (a et b) ou c
  - a et (b ou c)
  - a ou exclusif b
- Quelle est la valeur des expressions pour $x = 13$?

Tuples

Définition

- Type de données composé
- Stocker une collection d'éléments
- Non nécessairement de même type

lorsque que le premier élément est un atome, cet élément est un *marqueur*.

Synthaxe

- Eléments placés entre accolades
- Eléments séparés par des virgules

Tests immédiats

$\{123, bois\}.$
$\{un, \{22, police\}, 42\}.$
$\{\}.$
$\{person, 'James', 'Bond'\}.$
$\{person, 'Louis', 'Defunes'\}.$

Fonctions pré-définies sur les tuples

- `tuple_size/1` : nombre d'éléments du tuple
- `element/2` : obtenir un élément
- `setelement/3` : obtenir un tuple dont un élément est modifié

Tests immédiats

`tuple_size`$(\{\text{un}, \{22, \text{police}\}, 42\})$.
`tuple_size`$(\{\})$.
`element`$(2, \{\text{un}, \{22, \text{police}\}, 42\})$.
`setelement`$(2, \{\text{un}, \{22, \text{police}\}, 42\}, \text{deux})$.
$\{1, 2\} < \{1, 3\}$.
$\{1, 2\} < \{1, 2\}$.
$\{1, 2\} == \{1, 3\}$.

Listes

Ce n'est qu'un début...

Définition

- Type de données composé
- Stocker une collection d'éléments
- Non nécessairement de même type
- Les traitements ne sont pas les mêmes que sur les tuples (voir la prochaine séance !)

Synthaxe

- Eléments placés entre crochets
- Eléments séparés par des virgules

Tests immédiats

[lundi, mardi, mercredi, jeudi].
[456, cerise].
[].
[a, [b, [c, d, e],f], g].
Combien d'éléments dans cette dernière liste ?

Chaine de caractères

Caractère

- Représenté par un entier
- Obtenu en faisant précéder du caractère $

Chaine de caractères

- Pas de type spécifique aux chaines de caractères
- chaine de caractères : liste de code ASCII

Tests immédiats

$A.
$A +32.
$a.
[72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100].

Entrée / sortie

Entrées

`io:get_line/1` lit une ligne sur l'entrée standard, paramètre chaine d'invite
`io:get_chars/2` lit plusieurs caractères, param chaine d'invite et nombre de caractères
`io:read/1` lit un terme Erlang

Tests immédiats

`io:get_line`("une ligne svp :").
`io:get_chars`("3 caractères svp :", 3).
`io:read`("un terme erlang :").
Tester les conversions de type `list_to_integer/1` et `integer_to_list/1`.

Entrée / sortie

Sortie

`io:format/2` écrit de manière formatée sur la sortie standard.
paramètres : chaine de caractères et liste de valeurs à afficher

Formats possibles :

- ~c : caractère
- ~f : flottant avec 6 décimales
- ~e : flottant en écriture scientifique
- ~w : terme quelconque
- ~B : entier en base 10
- ~n : retour à la ligne

Exercice

Afficher le nombre de jours depuis votre naissance.

Pattern matching

Définition

- Affecter des valeurs à des variables
- Contrôler le flux d'exécution
- Extraire des valeurs à des données de type composé

Synthaxe

```
Motif = Expression
```

- `Motif` : expression composé de variables libres ou liées, ainsi que des valeurs littérales (atomes, entiers ou chaines)
- `Expression` : expression composé de structures de données, de variables liées, d'opérateur arithmétiques et d'appel de fonctions. Mais pas de variables libres.

Pattern matching

Déroulement

Exécute l'expression à droite de = puis compare la valeur au motif

- expression et motif doivent avoir la même forme
- les littéraux doivent être égaux aux valeurs de l'expression
- si le matching réussit, les variables libres sont liées aux valeurs correspondantes
- les variables liées doivent avoir les mêmes valeurs que dans l'expression

Exécution

2 résultats possibles lors de l'exécution :

- S'il a réussit, les variables libres sont liées et le résultat est celui de l'expression
- S'il a échoué, aucune variable libre n'est liée

Pattern matching

Test immediat

$\{ X, Y , 2 \} = \{12, 3, 2\}$
$\{ X, Y , 2 \} = \{12, 3, 2, 5\}$
$A = A + 1.$
$4 = 2 + 2.$
$3 = 2 + 2.$
$\{ \text{Element, Element, X} \} = \{1, 1, 12\}$
$\{ \text{Element, Element, X} \} = \{1, 2, 12\}$
$\{ \text{person, Name, Surname} \} = \{ \text{person, "Joys", "James"} \}$

_ est une variable "bidon" qui marque seulement l'emplacement

Test immediat

$[A, \_, [B, \_], \{B\}] = [1, 24, [45, 2], \{45\}]$

Module

On ne peut écrire de fonction directement dans le shell...

Définition

- Un module (fichier) regroupe des fonctions
- Le nom du fichier doit être celui du module avec l'extension `.erl`
- Le module doit contenir 2 directives :
  - `-module/1` pour le nom du module
  - `-export/1` pour exporter la liste des fonctions à exporter en dehors du module
- Les commentaires (jusqu'à la fin de la ligne) sont marqués par le %

```
-module(test).
-export([add/2]).

add(X, Y) ->
       X + Y.
```

Compilation

- La compilation produit un fichier `.beam` qui est exécutable par une machine abstraite (Björn's Erlang Abstract Machine)
- la fonction `cd/1` permet de changer de répertoire
- la fonction `c/1` permet de compiler

Test immédiat

```
-module(test).
-export([add/2]).

add(X, Y) ->
       X + Y.
```

Puis compiler et exécuter par :

```
test:add(3,4).
```

Fonction

Fonction

- L'entête est composée d'un nom (atome), suivi de paramètres entre parenthèses, puis de ->
- Fonction : ensemble de clauses séparées par des points-virgules
- Corps de chaque clause : plusieurs expressions séparées par des virgules

Test immédiat

```
area({square, Cote}) ->
   Cote * Cote ;
area({circle, Rayon}) ->
   math:pi() * Rayon * Rayon ;
area({triangle, A, B, C}) ->
   S = (A + B + C) / 2,
   math:sqrt(S * (S - A) * (S - B) * (S - C)) ;
area(Other) ->
   {error, invalid_object}.
```

Fonction

Exercice

• Ecrire un module `bool` qui contient des fonctions qui calculent les fonctions logiques *non*, *et*, *ou*, *nand*.

Bien sùr, vous n'êtes pas autorisé à utiliser les fonctions pré-définies.

Structure case

- Utilise le pattern matching
- Evaluation tour à tour jusqu'à trouver une correspondance
- le résultat est alors la dernière expression évaluée

```
case expression-conditionnelle of
  Motif1 -> expression1, expression2 ;
  Motif2 -> expression3, expression4 ;
   ... ;
  Motifn -> expressionk, expressionfin
end
```

Test immédiat

```
case X of
   0 -> toto ;
   1 -> un ;
   2 -> deux
end.
```

Structure if

- Ressemble à une instruction case

```
if
  Garde1 -> expression1, expression2 ;
  Garde2 -> expression3, expression4 ;
   ... ;
  GardeN -> expressionk, expressionfin
end
```

Les expressions "Garde" sont évaluées successivement jusqu'à une soit vraie

Test immédiat

```
if
   X > 0 -> positif ;
   X < 0 -> negatif ;
   X == 0 -> nulle
end
```

Sélecteur when

- une garde formée avec le sélecteur `when` suivie d'une autre garde : Condition supplémentaire à vérifier pour sélectionner la clause

Test immédiat

```
myFonc(N) when N > 0 ->
   positif ;
myFonc(0) ->
   nulle.
```

Structure conditionnelle

Exercice

- Ecrire de 3 manières une fonction qui calcule la parité d'un nombre entier

Exercice

Jeu didactique

Programmer le jeu qui consiste à deviner un nombre entre 1 et 100.