

ERLANGEN

Treball Dirigit

Llenguatges de programació

2021-2022

DidadoY

Índex

Introducció	3
Principals característiques	4
Paradigmes de programació	4
Programació funcional	4
Programació concurrent i distribuïda	4
Hot Swap	5
Sistema d'execució	5
Sistema de tipus	6
Assignació única	6
Tipat dinàmic	6
Principals aplicacions	7
Comparacions	8
Exemples de codi	8
Exemple de Quick Sort	8
Exemple de xifrat Caesar	9
Conclusions	9
Bibliografia	10
Descripció de les fonts d'informació	11
Avaluació de les fonts d'informació	11

Introducció

Erlang és un llenguatge de programació de propòsit general que va ser creat l'any 1984 per Ericsson, d'aquí prové el seu nom **Ericsson Language**. La idea principal era crear un llenguatge que fos capaç d'executar programes concurrentment de forma indefinida i per això es van agafar referències d'altres llenguatges com *Lisp*, *Haskell* i *Prolog*, intentant suprimir aquelles característiques que no es consideraven bones. Els processos d'*Erlang* són lleugers, independents i no pertanyen al sistema operatiu, pertanyen al mateix llenguatge.

Inicialment, Erlang, va ser concebut per aportar una millor manera de programar aplicacions de telefonia, cosa que té sentit, ja que la majoria d'aplicacions mòbils són altament concurrents, també explicarem altres característiques interessants que fan ideal aquest llenguatge per programar aplicacions de missatgeria que utilitzem en el nostre dia a dia.

Principals característiques

Paradigmes de programació

Erlang és conegut per ser un llenguatge multiparadigma, per tant, això significa que permet escriure programes fent servir diversos estils de programació. Seguidament, donarem pas als seus paradigmes funcional, concurrent i distribuït:

Programació funcional

El codi s'estructura en mòduls que contenen funcions i el subconjunt de programació seqüencial d'Erlang és un llenguatge funcional amb avaluació estricta. Per exemple, podríem fer el càlcul del factorial d'aquesta manera en Erlang:

```
-module(math).  
-export([factorial/1]).  
factorial(0) -> 1;  
factorial(N) when N > 0 -> N * factorial(N-1).
```

Fig.1: Exemple de programació funcional en Erlang.

Programació concurrent i distribuïda

Una de les majors virtuts d'Erlang és el suport de la concurrència de processos, tenint un petit conjunt de primitives que donen suport a la creació i comunicació de processos. Els processos poden ser creats de manera local al mateix node o donant pas a una xarxa de nodes en sistemes distribuïts, per tant, Erlang té la capacitat d'executar processos en més d'un node diferent i es segueixen comunicant com si estiguessin en el mateix node.

Un petit exemple en seria aquest de l'ús de les primitives del llenguatge en seria aquest:

```
Pid = spawn(Mod, Func, Args)      % ejecutar la función Func como un nuevo proceso
Pid = spawn(Node, Mod, Func, Args) % ejecutar la función Func en un nodo remoto

Pid ! a_message                   % enviar un mensaje al proceso (asíncronamente)

receive                           % recibir el mensaje enviado a este proceso
  a_message -> do_something
end.
```

Fig.2: Exemple de creació de processos d'Erlang

La principal forma de control d'errors d'Erlang també es basa en la concurrència.

Hot Swap

Una d'entre altres característiques que fa molt interessant aquest llenguatge de programació és el *Hot Swap*. Aquest mecanisme permet actualitzar el codi de l'aplicació sense canviar el seu estat, cosa que permet actualitzar una aplicació sense fer manteniment, ja que el codi nou substitueix el vell i poden coexistir sense alterar l'estat d'execució.

Sistema d'execució

Erlang té un sistema d'execució que conté una màquina virtual anomenada BEAM on es poden executar els arxius un cop compilats, també inclou un sistema de llibreries OTP (*Open Telecom Platform*). Els arxius programats en Erlang tenen l'extensió *.erl* la qual permet la compilació dels mòduls utilitzats en el codi i obtenint així un fitxer *.beam*, aquest arxiu es pot executar a la màquina virtual que hem esmentat per provar el funcionament del codi.

El sistema d'execució d'Erlang està pensat per sistemes amb aquestes característiques:

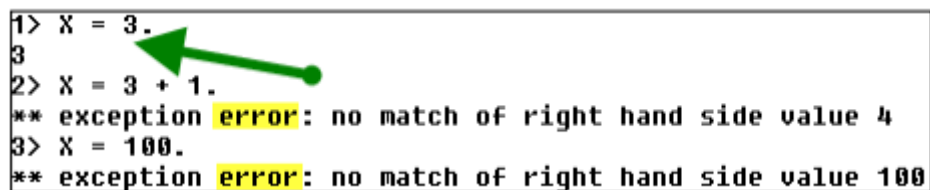
- Sistemes distribuïts
- Sistemes que puguin permetre's fallar
- Sistemes que hagin d'estar executant-se 24/7
- Sistemes que hagin d'actualitzar-se sense parar el seu funcionament (*Hot Swapping*).

Sistema de tipus

Una de les grans diferències que es troba en Erlang respecte a altres llenguatges és el seu sistema de tipus el qual té les següents característiques.

Assignació única

L'assignació de variables és única això vol dir que no podem canviar el valor de la variable, no se'ns permet fer $X=X+1$ com feiem a C++. Aquí en tenim un exemple de què passa si intentem canviar el valor:



```
1> X = 3.
3
2> X = 3 + 1.
** exception error: no match of right hand side value 4
3> X = 100.
** exception error: no match of right hand side value 100
```

Fig.3: Exemple de error a l'assignar més d'un valor a una variable

Tipat dinàmic

No fa falta especificar el tipus d'una variable, Erlang s'encarrega de detectar el tipus dinàmicament. En té els següents tipus de dades i estructures:

- **Boolean.**
- **Integers:** De mida ilimitada. `1212232212`
- **Float:** `12.42342`
- **String:** Els *strings* es representen entre cometes. *"Això és un string"*
- **Atom:** Són **constants** amb nom i els podem declarar de dues maneres, o començant en minúscula, `aixo_es_un_atom`, o entre cometes úniques, `'Això també és un àtom'`
- **List:** Les llistes poden contenir molts tipus alhora. `[12, "abc", elmeuatom]`
- **Tuples:** Les estructures de dades exactament com les llistes, `{12, "abc", elmeuatom}`
- **Binaries:** Emmagatzema un conjunt de bits múltiples de 8. `<<"abc">>`
- **Pids:** Permet crear un nou procés. `spawn(...)`
- **Maps:** `{Key1=>Value1,...,KeyN=>ValueN}`
- **Funcions:** `funcio(...) -> -> end`

Principals aplicacions

Com ha estat esmentat en aquest document, Erlang neix de la necessitat d'utilitzar un llenguatge que faciliti la programació d'aplicacions de telefonia. Per tant, ara introduïrem aplicacions que usen Erlang per a facilitar la manipulació concurrent de processos.

Un clar exemple d'aquest mètode n'és Whatsapp. Whatsapp usa aquest llenguatge per executar els servidors de missatgeria, aconseguint mantenir aproximadament 2 milions d'usuaris alhora per cada servidor. Podem observar també que Whatsapp compleix els requeriments per utilitzar Erlang sent aquest una aplicació que necessita un manteniment sense aturar el servei fent ús del Hot Swap esmentat abans aconseguint així que Whatsapp no s'aturi mai quan és actualitzat permetent conviure el codi antic amb el codi nou.

Unes altres grans companyies que utilitzen *Erlang* a diari son les següents:

- **Amazon:** Utilitza Erlang per implementar la *SimpleDB*
- **Yahoo!:** Utilitza Erlang per mantenir els marcadors dels usuaris guardats sense perdre les dades.
- **Facebook:** Utilitza *Erlang* per potenciar el seu servei *backend* de xat
- **T-Mobile:** Utilitza Erlang per l'enviament d'SMS i sistemes d'autenticació concurrents

Més exemples de grans companyies poden ser afegides per utilitzar *Erlang* en els seus productes, podem veure que a pesar de ser un llenguatge antic, encara es segueix utilitzant en aplicacions de missatgeria instantània contemporànies.

Comparacions

Hem pogut observar que *Erlang* és un llenguatge molt similar a *Haskell* i a altres llenguatges de programació funcional si de sintaxi en parlem, una de les diferències notables a l'hora de programar en *Erlang* és la seva estructura de mòduls del llenguatge.

També, *Erlang* en tenir com a referència *Prolog*, poden assemblar-se sintàcticament encara que *Erlang* va suprimir el retrocés que en tenia *Prolog* a l'hora d'unificar, quan fallava desfeia l'acció i n'agafava una de nova, però a *Erlang* van decidir no aplicar aquest mètode fent ús de *l'elecció compromesa* on un cop escollies que havies de fer, no podràs tornar enrere.

Exemples de codi

A continuació exposarem petits exemples de codi pero ajudar a comprendre millor el llenguatge:

Exemple de Quick Sort

Podem observar que és molt similar a altres llenguatges de programació funcional. En aquest codi la funció *qsort* ordena una llista de més petit a més gran.

```
% This is file 'listsrt.erl' (the compiler is made this way)
-module(listsort).
% Export 'by_length' with 1 parameter (don't care about the type and name)
-export([by_length/1]).

by_length(Lists) -> % Use 'qsort/2' and provides an anonymous function as a parameter
    qsort(Lists, fun(A,B) -> length(A) < length(B) end).

qsort([], _) -> []; % If list is empty, return an empty list (ignore the second parameter)
qsort([Pivot|Rest], Smaller) ->
    % Partition list with 'Smaller' elements in front of 'Pivot' and not-'Smaller' elements
    % after 'Pivot' and sort the sublists.
    qsort([X || X <- Rest, Smaller(X,Pivot)], Smaller)
    ++ [Pivot] ++
    qsort([Y || Y <- Rest, not(Smaller(Y, Pivot))], Smaller).
```

Fig.4: Exemple de quicksort

Exemple de xifrat Caesar

En aquest codi podem observar un exemple de xifrat Caesar on li suma 13 a la posició de les lletres d'una llista per obtenir una llista xifrada amb totes les posicions de les lletres de l'abecedari mogudes 13 posicions. Restarem 13 pel cas contrari a l'hora de descifrar el codi caesar.

```
-module (cesar).  
-export ([cifrar/1, descifrar/1]).  
  
cifrar(String) ->  
    list_to_atom([X+13 || X <- String]).  
  
descifrar(String) ->  
    list_to_atom([X-13 || X <- String]).
```

Fig.5: Exemple de xifrat Caesar

Conclusions

Després de la realització d'aquest treball, he comprovat que Erlang és un llenguatge molt útil per aquelles aplicacions que necessiten ser actualitzades, però sense aturar el seu funcionament, per tant, m'ha sorprès moltíssim el Hot Swap, ja que ho trobo una de les millors característiques que té aquest llenguatge de programació a part de la seva característica principal que és l'execució de moltíssims processos independents concurrents, sent una característica molt útil per tots aquells dispositius que necessiten mantenir molts usuaris alhora executant processos totalment diferents.

Des de el meu punt de vista, Erlang, malgrat ser un llenguatge de programació bastant antic, he pogut observar que es segueix utilitzant avui dia en aplicacions que utilitza una gran part de la població mundial i, per tant, avui dia segueix sent un llenguatge amb moltíssim potencial que pot seguir aplicant-se en futures aplicacions que necessitin uns alts requisits que Erlang és capaç de suportar.

Bibliografia

- [1] Universitat Politècnica de València. (2015). Diseño de aplicaciones multi-paradigma en Erlang. <https://riunet.upv.es/bitstream/handle/10251/68592/MUÑOZ%20-%20Diseño%20de%20aplicaciones%20multi-paradigma%20en%20Erlang.pdf?sequence=2>
- [2] Montenegro, M. (2016). Perdiendo el miedo a la programación concurrente. Erlang. <https://informatica.ucm.es/data/cont/media/www/pag-118038/Erlang.pdf>
- [3] Universitat de Girona. (2009). Implementació d'un sistema xat multiprotocol fent servir el llenguatge de programació ERLANG. <https://dugi-doc.udg.edu/bitstream/handle/10256/1860/2%20Memòria.pdf?sequence=2&isAllowed=y>
- [4] Erlang -- Data Types. (20019). Erlang. http://erlang.org/documentation/doc-6.0/doc/reference_manual/data_types.html
- [5] EcuRed. (2020). Erlang - EcuRed. Erlang. <https://www.ecured.cu/Erlang>

Descripció de les fonts d'informació

Per aquest treball he decidit buscar informació en articles acadèmics o material d'altres universitats com bé he referenciat a la bibliografia. En aquest cas he trobat molta informació sobre el llenguatge *Erlang* a documents de la *Universitat Politècnica de València* i la *Universitat de Girona*. Per trobar més informació sobre els tipus de dades del llenguatge he decidit consultar el manual oficial del llenguatge. He procurat evitar fonts d'informació no fiable que no provenguin de centres educatius. La informació la he anat buscant segons l'apartat en el que estava en comptes de generalitzar la búsqueda sobre *Erlang* per així obtenir resultats més precisos.

Avaluació de les fonts d'informació

Les fonts que he utilitzat són organitzacions educatives o la pàgina oficial d'Erlang, per tant, no existeix la possibilitat d'obtenir informació no vàlida d'aquest document. He procurat no utilitzar *Wikipedia* com font d'informació per aquest document encara que *Wikipedia* sí que serveix per introduir-se i posar-te en contexte abans de començar a redactar. Per altre part, la informació que hi ha en aquest document es pot comprovar fàcilment utilitzant el llenguatge o anant directament a la web oficial del llenguatge de programació.