



Pràctica 2: Erlang: funcions d'ordre superior i concurrència

Programació Concurrent i en Temps Real — iTIC

Antoni Escobet Canal
Sebastià Vila-Marta

26 de juliol de 2018

Índex

1 Organització	1
1.1 Lliurament	1
2 Exercicis	1

1 Organització

Aquesta sessió s'organitza com una seqüència de problemes de dificultat creixent l'objectiu dels quals és implementar petits programes escrits en Erlang. Amb l'objectiu de reforçar l'hàbit d'usar sistemes de control de versions, cal desenvolupar la pràctica amb el suport del sistema que ofereix <http://escriny3.epsem.upc.edu>.

1.1 Lliurament

Cal lliurar els exercicis en un tarfile a través d'Atenea en la data fixada. Cal que el desenvolupament es faci usant Subversion a través de les facilitats que ofereix <http://escriny3.epsem.upc.edu>.

2 Exercicis

EXERCICI 2.1 Implementeu la funció d'ordre superior `fold(L,F,I)` que té com a paràmetres una llista `L`, una funció de dues variables `F` i un valor inicial `I`. El seu funcionament és tal que si $L = l_0, l_1, l_2, \dots, l_n$, llavors calcula l'expressió $F(\dots F(F(I, l_0), l_1) \dots, l_n)$.

Useu aquesta funció per definir una funció tal que, donada una llista d'enters en calcula el producte. Quins altres problemes classics podríeu resoldre amb la funció `fold`?

EXERCICI 2.2 Estudieu amb atenció les funcions predefinides `lists:map`, `lists:all`, `lists:any` i `lists:filter`. Juntament amb `lists:foldl` i `lists:flatten` són un conjunt de funcions d'ordre superior molt corrents en els llenguatges funcionals.

Usant aquestes funcions definiu noves funcions que calculin:

1. `sumValSenar`: La suma dels valors senars d'una llista d'enters. $L=[2,5,7,8,10]$; $5+7=12$.
2. `sumPosPar`: La suma dels valors que ocupen posicions parells en una llista d'enters. $L=[3,2,4,6]$; $2+6=8$.
3. `dupValPar`: La llista obtinguda duplicant els elements de les posicions parells de la llista original. $L=[3,2,6]$; $[3,2,2,6]$.
4. `valUnPos`: Un predicat sobre una llista que valgui cert només si la llista conté un sol valor positiu. $L=[-1,4,-3]$; cert.

EXERCICI 2.3 Una forma d'accelerar el càlcul del producte escalar quan tenim vectors de dimensió molt elevada és subdividir els vectors per la meitat, calcular el producte escalar per a cadascun dels vectors en un procés independent i sumar els productes parcials obtinguts dels processos. Aquest és l'objectiu d'aquest exercici.

Seguiu els següents passos per implementar la solució:

1. Definiu una funció tal que donades dues llistes (vectors) calcula el producte escalar.
2. Definiu una funció que defineix un procés calculador de productes escalars. Aquesta funció rep tres paràmetres: les dues llistes que corresponen als vectors i l'identificador del procés al que cal retornar el resultat.
3. Finalment definiu la funció `pe()`, que rep com a paràmetres dues llistes, les subdivideix per la meitat (useu la funció `lists:split()`), arrenca dos processos de càlcul concurrents cadascun amb una de les meitats del problema i espera a que tornin el resultat.

Per demostrar que els processos s'executen correctament, podeu utilitzar la funció de depuració de l'Erlang. Per usar-la, s'ha de compilar el mòdul amb l'opció: `debug info`

`c(nom_modul, [debug_info])`.

I cridar el depurador amb la comanda: `debugger:start()`.

Busqueu informació sobre aquest depurador per poder fer les comprovacions oportunes.

EXERCICI 2.4 Seguint la idea emprada en l'exercici anterior implementeu una funció que ordeni una llista usant dos subprocessos. L'estratègia consisteix a subdividir la llista original en dues subllistes, ordenar-les en sengles subprocessos usant la funció `lists:sort()`, i fusionar les subllistes ordenades obtingudes usant la funció `lists:merge()`.