Conceptos y Modelos de Programación
Paso de mensajes, 2006

Camilo Rueda [1]

[1] Universidad Javeriana-Cali

16 de marzo de 2007

Control de errores en paso de mensajes

```
proc {ServerProc Msg}
  case Msg
  of sqrt(X Y E) then
    try
     Y={Sqrt X}
     E=normal
    catch Exc then E = exception(Exc) end
  end
end
Server={NewPortObject2 ServerProc}
```

Invocación

- Síncrono: la invocación en el cliente es

  {Send Server sqrt(X Y E)}
  case E of exception(Exc) then raise Exc end end

  El cliente se bloquea hasta que E se ligue

- Asíncrono
  Un hilo en el chequeo de la excepción en el cliente

Uso directo de paso de mensajes

- Estructuras de datos modificables concurrentemente
  - Un puerto para cada mensaje

```
fun {CrearCola}
   Recibir PuertoRec = {NewPort Recibir}
   Dar PuertoDar = {NewPort Dar}
   in
   Recibir = Dar
cola(put : proc {$ X} {Send PuertoDar X} end
     get : proc {$ X} {Send PuertoRec X} end)
end
```

Problemas?

Uso directo de paso de mensajes(2)

```
fun {CrearCola}
   Recibir PuertoRec = {NewPort Recibir}
   Dar PuertoDar = {NewPort Dar}
   proc {Match Xs Ys}
      case Xs # Ys
      of (X | Xr)#(Y|Yr) then X = Y {MatchXr Yr}
      [ ] nil#nil then skip
      end
   end
   in
   thread {Match Dar Recibir} end
   cola(put : proc {$ X} {Send PuertoDar X} end
        get : proc {$ X} {Send PuertoRec X} end)
end
```

Por qué el hilo?

Detección de terminación

- un hilo puede crear sub-hilos
- El hilo puede requerir saber cuándo terminan sus subhilos
- Cómo hacer esto de manera encapsulada?
  - Crear un nuevo Hilo:
    $\{NewThread\ P\ SubThread\}$
  - Dentro de P los hilos se crean invocando
    $\{SubThread\ P1\}$

Hay tres maneras entonces de crear hilos:

thread *Instruccion* *end*
$\{NewThread\ proc\ \{\$\}\ instr\ end\ ?SubThread\}$
$\{SubThread\ proc\ \{\$\}\ instr\ end\}$

implementación

```
local
   proc {ZeroExit N Is}
      case Is of I | Ir then
         if N + I\ = 0 then {ZeroExit N + I Ir} end
      end
   end
in
   proc {NewThread P ?SubThread}
      Is   Pt = {NewPort Is}
      in
      proc {SubThread P}
         {Send Pt 1} thread {P} {Send Pt  - 1} end
      end
         {SubThread P}    {ZeroExit 0 Is}
      end
end
```