Paradigmas de la Programación
Práctico 7: Concurrencia

Laura Alonso Alemany Cristian Cardellino Ezequiel Orbe

2 de junio de 2015

1. En el siguiente ejemplo se está usando la variable común `lock` como lock. Describa una ejecución del código que lleve a un estado inconsistente, describiendo en orden las asignaciones y evaluaciones que ocurren al ejecutar el código. Explique cómo se podría evitar este funcionamiento no deseado con implementaciones atómicas de `wait` y `signal`.

```
lock := 0;
cobegin
  begin
    while lock=1 do end;  // loop que no hace nada hasta que el
       lock sea 0
    lock:=1;              // setear el lock para entrar a la
       seccion critica
    sign up(fred);        // seccion critica
    lock:=0;              // soltar el lock
  end;
  begin
    while lock=1 do end;   // loop que no hace nada hasta que
       el lock sea 0
    lock:=1;               // setear el lock para entrar a la
       seccion critica
    sign up(bill);         // seccion critica
    lock := 0;             // soltar el lock
  end;
end;
```

2. En java existen dos formas de crear un nuevo *thread* explícitamente: creando una subclase de la clase `Thread` o bien implementando la interfaz `Runnable`. En ambos casos el nuevo *thread* debe implementar el método `run`, que contendrá el código ejecutado por el *thread*. Cuál de estas dos opciones crea un objeto más versátil y por qué?

3. El siguiente es un programa concurrente en Erlang. Escriba su versión secuencial.

```
f1(0) ->
    0;
f1(1) ->
    1;
f1(N) ->
    Self = self(),
    spawn(fun() ->
                  Self ! f0(N-1)
          end),
    spawn(fun() ->
```

```
                    Self ! f0(N-2)
            end),
    receive
        F1 ->
            receive
                F2 ->
                    F1 + F2
            end
    end.
   }
 }
```

4. Los actores se comunican mediante mensajes desordenados. Explique qué información y métodos debería añadir a un modelo basado en actores para poder procesar los mensajes recibidos en el orden en el que fueron enviados.

5. Explique por qué el esquema productor - consumidor no tiene condiciones de carrera.

6. En lenguajes con evaluación perezosa la concurrencia sin condiciones de carrera puede implementarse mediante estructuras de datos. Explique cómo, a partir del siguiente ejemplo:

```
lista_de_dobles = map (\x -> 2 * x) lista_de_todos_los_naturales
```

7. Hay ocho combinaciones posibles en el pasaje de mensajes entre procesos concurrentes, combinando sincronicidad / asincronicidad, ordenado / no ordenado y con buffer / sin buffer. Explique cuáles tienen ventajas y cuáles no tienen sentido o tienen desventajas.