Programación de Sistemas Concurrentes y Distribuidos 2ª Convocatoria curso 15/16

8 de septiembre de 2016

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Ejercicio 1 (2.5 ptos.)

Considérese el siguiente programa concurrente, en el que tres procesos iteran escribiendo distinta información por la salida estándar. Se pide completar el código para que, usando monitores, se cumplan simultáneamente los invariantes siguientes:

1. `num(A) >= num(C)+num(E)`
2. `num(D) >= num(F)`
3. `num(E) >= num(F)`

donde, para un carácter dado <c>, `num(c)` representa el número de veces que se envía a la salida estándar.

```
process P1              process P2              process P3
   while TRUE              while TRUE              while TRUE
      write("A")              write("C")              write("E")
      write("B")              write("D")              write("F")
   end while               end while               end while
end process             end process             end process
```

Ejercicio 2 (2.5 ptos.)

Se trata de resolver el mismo problema anterior, pero esta vez utilizando semáforos como mecanismo para la sincronización.

Ejercicio 3 (2 ptos.)

En este ejercicio se trata de aplicar el algoritmo de los Generales Bizantinos a dos escenarios diferentes.

1. El primer escenario consta de 4 generales (Basilio, Zoe, León y Juan), donde 2 generales deciden "atacar" (Basilio y Juan) y los otros dos "retirarse" (Zoe y León). Además, Juan es un general traidor que experimenta un fallo por caída en segunda ronda. Antes de que se produzca este fallo, Juan informa al resto de los generales qué decisión tomó Basilio en primera ronda, y también le comunica a Zoe la decisión de León.

   Se pide:

   - Detallar el resultado de aplicar el algoritmo para el escenario descrito, rellenando las tablas de decisión de los 4 generales involucrados (se adjuntan como parte del enunciado unas tablas vacías) y explicando de forma breve los mensajes intercambiados en primera y segunda vuelta entre estos generales.
   - Sobre la solución propuesta, ¿cuál es la decisión final de los generales leales? y ¿son consistentes sus decisiones? Responde de manera razonada y concisa a estas preguntas.

2. El segundo escenario también está compuesto por cuatro generales (Basilio, Zoe, León y Juan). Uno de estos generales es traidor y genera fallos bizantinos en segunda vuelta con el propósito de provocar inconsistencias en las decisiones de los generales leales. El fallo concreto consiste en comunicar la decisión contraria de un general leal al resto de generales (por ejemplo, si un general leal decidió "atacar", el traidor informará al resto de generales que decidió "retirarse", o viceversa). La Figura 1 muestra la tabla de decisión de Zoe, uno de los generales leales, después de ejecutar el algoritmo.

   Se pide:

   - En base a la tabla de decisión de Zoe, ¿qué general es el traidor? y ¿qué decisión adopta originalmente cada general? Responde de manera razonada y concisa a la pregunta.
   - Describir un ejemplo concreto de aplicación del algoritmo que tenga como

Zoe					
General	**Plan**	**Reported by**			**Majority**
		Basilio	León	Juan	
Basilio	A	—	A	A	A
Zoe	A	—	—	—	A
León	R	R	—	R	R
Juan	R	R	A	—	R
Majority					R

Figura 1: Información de Zoe tras la segunda vuelta

resultado la tabla de decisión anterior de Zoe. La descripción debe detallar los mensajes concretos que intercambian los generales en primera y segunda vuelta, así como sus correspondientes tablas de decisión resultantes (para cumplimentar esta última información también se proporcionan adjuntas al enunciado las tablas vacías de los generales involucrados).

- Sobre el escenario planteado como solución al apartado anterior, ¿cuál es la decisión final de los generales leales? y ¿son consistentes sus decisiones? Responde de manera razonada y concisa a estas preguntas.

Ejercicio 4 (3 ptos.)

Un jardín tiene dos puertas de acceso. Cada puerta tiene instalado un torno responsable de gestionar las entradas y salidas del jardín. Debido a su reducido tamaño no puede haber más de 50 personas simultáneamente visitándolo. Inicialmente, hay 100 personas interesadas en realizar la visita. Cuando una persona quiere acceder al jardín decide la puerta concreta de entrada y, una vez finalizada la visita, sale por esa misma puerta.

Todas las personas se comportan de la misma manera, repitiendo la siguiente secuencia de acciones durante un número aleatorio de veces: primero, decide aleatoriamente la puerta de acceso; segundo, solicitan a la puerta asignada permiso para entrar en el jardín y esperan a que ésta les dé acceso; tercero, visitan el jardín durante un cierto tiempo; y, finalmente, solicitan salir y esperan a que el torno les autorice la salida. Por otro lado, el funcionamiento de los tornos es muy simple: reciben las solicitudes de los usuarios y les dan los permisos tanto de entrada como de salida.

Se pide implementar el sistema distribuido anteriormente descrito y explicar de forma concisa las principales decisiones de diseño adoptadas. El sistema consta como mínimo de dos tipos de procesos: los 2 procesos *puerta* y los 100 procesos *persona*. La comunicación entre los distintos procesos involucrados en el sistema estará basada en canales síncronos

(no hay memoria compartida). Además, por simplicidad, vamos a suponer que existe una función, llamada *decidirPuerta()*, que permite a un proceso persona decidir la puerta concreta de acceso al jardín para cada visita (esta función local a cada proceso puerta no es necesario programarla como parte de la solución).