Programación Concurrente

UCM – Facultad de Informática
Grado en Ingeniería Informática
2012-2013
Curso 3º A

Juan Luis Álvarez Herradón
Ángel Luis Ortiz Folgado

UNIVERSIDAD COMPLUTENSE
MADRID

[MEMORIA DE LA PRÁCTICA 3]

PREGUNTAS

EJERCICIO 1

Consideremos si esta solución es equitativa en el sentido siguiente: ¿Puede producirse la inanición? (es decir, la situación en que un filósofo pasa un tiempo infinito en el estado hambriento sin nunca llegar a comer)? Explique cómo.

Al principio si se puede producir inanición ya que se puede dar el caso que haya dos vecinos de un comensal que se compinchen de tal forma que cuando uno deja el tenedor el otro coge el otro tenedor dejando al comensal central en inanición. El que lo deja no lo deja hasta que está seguro que el otro lo puede coger.

EJERCICIO 3

Explique muy brevemente cómo funciona la notificación optimizada en su solución. ¿Puede producirse la inanición? Explique cómo.

Para la implementación del monitor básico, se utiliza el monitor propio de la clase MonitorBasico y una variable que indica si hay algún escritor escribiendo. Además se lleva la cuenta del número de lectores que han tomado el control para leer el dato. Así cuando se termina de leer el dato, se puede avisar al escritor que se ha terminado.

El problema surge porque los lectores acaparan toda la actividad y no permiten al escritor escribir, ya que los escritores se quedan esperando a que todos los lectores terminen, pero éstos a medida que terminan, vuelven a querer leer.

Por lo que se produce la inanición de los escritores.

EJERCICIO 4

Explique muy brevemente cómo su solución consigue dar prioridad a los lectores, mencionando cualquier suposición que necesite hacer sobre la planificación de hilos en la máquina virtual de Java. ¿Se puede optimizar la notificación al igual que en el ejercicio 1? ¿Puede producirse la inanición? Explique cómo.

En la segunda implementación agregamos una variable escritoresEnEspera que hace que los lectores se queden a la espera si hay algún escritor esperando a que los lectores terminen.

El problema ahora es que el escritor que primero tome el control hace que el resto de hilos sufran de inanición, ya que obtiene el control y el período por el que permite a otros hilos es demasiado corto, por lo que nunca logran entrar los demás.

EJERCICIO 5

Explique muy brevemente cómo su solución consigue la ausencia de inanición categórica, mencionando cualquier suposición que necesite hacer sobre la planificación de hilos en la máquina virtual de Java. ¿Puede producirse la inanición individual? Explique cómo.

Para la tercera mejora, añadimos una variable turnoLectores, que permite avisar a escritores que los lectores todavía no han leído el dato, por lo que se ponen a la espera y hasta que no hayan leído, no toma el control.

Así si bien los escritores tienen prioridad sobre los lectores, no se produce inanición categórica.

EJERCICIO 6

Explique muy brevemente cómo su solución consigue la ausencia de inanición individual.

En la última mejor, introdujimos un secuenciador que genera números de secuencia, de manera que cada hilo que desee entrar debe sacar un número y esperar a que un contador del monitor, llegue a su número.

Esto produce un comportamiento FIFO, y permite que no se produzca inanición, ya que cada uno tiene un turno y el resto que quiera ejecutar, va a sacar otro, que siempre llegará pase lo que pase.

Para permitir la concurrencia de los lectores, éstos avanzan el turno cuando les toca, mientras que los escritores lo hacen al terminar, lo cual permite que pueda haber más de un lector ejecutándose, si bien cuando hay un escritor escribiendo, nadie más lee ni escribe.