Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Asignación de Prácticas Número 4

Las primeras estrategias para el control de la exclusión mutua se basaron en algoritmos que compartían información utilizando varibles comunes. Ello permitía que si una hebra deseaba ejecutar su sección crítica, pudiera saber qué hacían los demás antes de acceder a ella, y esperar en caso negativo mediante un bucle de espera activa, en el cual se está comprobando continuamente el estado de esas variables hasta encontrar la situación adecuada para el ingreso en la sección crítica. En esta práctica se le pide, al objeto de que se familiarice con esta clase de algoritmos de control de e.m. y la estrategia que emplean, el análisis de varios de ellos, y también alguna implementación. **Documente su código con etiquetas**.

1. Ejercicios

1. Pida a ChatGPT información sobre el uso y funcionamiento de las variables cualificadas como *volatile* en Java.

2. Haga lo mismo con el método `Thread.yield()`.

3. Abra este URL: `https://github.com/motib/concurrent-distributed/tree/main/Java`. En él, encontrará buena parte de las soluciones teóricas para la programación concurrente que se proponen en el libro *Principles Of Concurrent and Distributed Programming* (M. Ben-Ari) programadas en Java. Descargue las cuatro etapas del refinamiento sucesivo (incorrectas) para una solución al problema de la exclusión mutua con variables comunes y esperada ocupada: ficheros `First.java, Second.Java, Third.Java` y `Fourth.java`.

4. Someta a análisis y ejecución cada una de ellas; para cada intento, tome nota del comportamiento observado, y razone por qué se produce.

5. Descargue ahora el fichero `Dekker.java`, y haga lo propio.

6. Genere una nueva versión del algoritmo de Dekker, y guárdela en un fichero `Dekker2.java` efectuando los siguientes cambios: sustituya los lazos

`while(true)` por bucles `for` con un número finito de iteraciones, y establezca un contador compartido para ambas hebras, cuyo valor inicial será cero, y que se incrementará en las respectivas secciones críticas (elimine de ellas las impresiones al terminal). Ahora, elabore una tabla de doble entrada, cuya primera columna será el número de iteraciones de los bucles `for`, que irán creciendo, y la segunda el valor **final** impreso obtenido para el contador.

7. En la página 68 del libro *Programación Concurrente*, 1ª edicion de 2003 de Palma et al. encontrará un análisis del algoritmo incorrecto de *Hyman*; escriba en `Hyman.java` su implementación del mismo, y construya una tabla como en el caso anterior.

8. El algoritmo de *Eisenberg-McGuire* proporciona exclusión mutua sobre variables comunes con espera ocupada. Pida a ChatGPT información sobre el funcionamiento del algoritmo, y también una implementación del mismo. Verifíquela informalmente.

9. Haga lo propio con el algoritmo de exclusión mutua de *Lamport*.

10. Redacte un corto documento utiizando Latex (vía Overleaf o instalación local) llamado analisis.pdf donde para cada código analizado, incluirá el comportamiento observado y la razón del mismo.