Programación Concurrente y de Tiempo Real*
Grado en Ingeniería Informática
Asignación de Prácticas Número 5

Las primeras estrategias para el control de la exclusión mutua se basaron en algoritmos que compartían información utilizando varibles comunes. Ello permitía que si un hilo 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 estregia que emplean, que realice la implementación de varios de ellos. Documente todo su código con etiquetas (será sometido a análisis con `javadoc`). Si lo desea, puede también agrupar su código en un paquete de clases, aunque no es obligatorio.

1. Ejercicios

1. Pruebe los algoritmos de control con variables comunes recogidos en el guión de prácticas número 5. Familiarícese con la técnica de control de la e.m. basada en el uso variables compartidas, con el uso en java de variables `volatile` y con las diferentes aproximaciones al algoritmo de *Dekker*. Modifique el fichero `Dekker.java` para que funcione con tres procesos. Guarde su trabajo en `algDekker.java` y escriba un corto documento `analisis.pdf` que recoja el comportamiento obtenido y su interpretación del mismo.

2. Escriba un programa que implemente el Algoritmo de *Eisenberg-McGuire* para dos procesos:

`http://en.wikipedia.org/wiki/Eisenberg_&_McGuire_algorithm`

Cree dos hilos (que deberán compartir las variables de control comunes y un recurso común) por implementación de la interfaz `Runnable`, y guarde su código en `algEisenbergMcGuire.java`. Utilice un ejecutor de tamao fijo. Dedique un apartado del documento `analisis.pdf` a este algoritmo que recoja el c omportamiento obtenido y su interpretación del mismo.

3. Una solución válida para el problema de la exclusión mutua con n procesos es el algoritmo de la panadería de *Lamport*. Ver

`http://es.wikipedia.org/wiki/Algoritmo_de_la_panader%C3%ADa_de_Lamport`

Escriba un programa[1] `algoLamport.java` que lo implemente para n procesos utilizando objetos que implementen la interfaz `Runnable` y un ejecutor de tamao fijo.

2. Procedimiento y Plazo de Entrega

Se ha habilitado una tarea de subida en *Moodle* que le permite subir cada fichero que forma parte de los productos de la práctica de forma individual en el formato original. Para ello, suba el primer fichero de la forma habitual, y luego siga la secuencia de etapas que el propio *Moodle* le irá marcando. Recuerde además que:

- No debe hacer intentos de subida de borradores, versiones de prueba o esquemas de las soluciones. *Moodle* únicamente le permitirá la subida de los ficheros por **una sola vez**.
- La detección de plagio (copia) en los ficheros de las prácticas, o la subida de ficheros vacíos de contenido o cuyo contenido no responda a lo pedido con una extensión mínima razonable, invalidará plenamente la asignación, sin perjuicio de otras acciones disciplinarias que pudieran corresponder.
- El plazo de entrega de la práctica se encuentra fijado en la tarea de subida del Campus Virtual.
- Entregas fuera de este plazo adicional no serán admitidas, salvo causa de fuerza mayor debidamente justificadas mediante documento escrito.
- Se recuerda que la entrega de todas las asignaciones de prácticas es recomendable, tanto un para un correcto seguimiento de la asignatura, como para la evaluación final de prácticas, donde puede ayudar a superar esta según lo establecido en la ficha de la asignatura.

[1] En la carpeta de código de la práctica se le proporciona una versión para dos procesos.