Programación Concurrente y de Tiempo Real*
Grado en Ingeniería Informática
Asignación de Prácticas Número 8

La presente asignación tiene por objetivo habituarle al desarrollo de soluciones de control de la concurrencia basadas en monitores teóricos, y en su implementación posterior utilizando el lenguaje java y las primitivas de control que este provee para ello en el API estándar para control de concurrencia. 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. Enunciados

1. Se dispone de n procesos concurrentes que comparten tres dispositivos de impresión. Antes de poder utilizarlos, un proceso debe pedir que le asigne una de las impresoras que esté libre, si la hay. En caso contrario, deberá esperar. En caso de que la haya, el sistema asignará una impresora disponible al proceso peticionario. Cuando la tarea de impresión concluye, el proceso debe liberarla explícitamente. Se pide:

- Desarrolle un monitor teórico que modele el sistema, utilizando variables de condición y disciplina de señalización `SC` (señalar y seguir). Guarde su solución en un documento `monitorImpresion.pdf`. Deberá desarrollar el documento utilizando LaTeX con *OverLeaf*.
- A partir del monitor anterior, efectúe su implementación en lenguaje Java, aplicando el protocolo explicado en clase de teoría y encapsulando los recursos necesarios, escribiendo el conjunto de métodos que proporcionen la gestión desarrollada en el monitor teórico, y sincronizando cuando sea necesario. Guarde el código del monitor en `monitorImpresion.java`, y escriba ahora un programa que lo utilice donde haya múltiples hilos concurrentes. Guarde este programa en `UsamonitorImpresion.java`.

2. Suponga ahora que dispone de una cadena de producción formada por tres procesos que realizan las siguientes tareas:

- `procesoA`: genera matrices cuadradas aleatorias de $10^4 \times 10^4$ de números enteros. Las matrices generada por este proceso se almacenarán en un `buffer_1` de matrices de hasta 100 ranuras.

- `procesoB`: toma desde el `buffer_1` anterior las matrices generadas por el `procesoA` y realiza su transposición, para luego depositarlas en un `buffer_2` de hasta 50 ranuras.
- `procesoC`: toma desde el `buffer_2` anterior las matrices transpuestas generadas por el `procesoB`, realiza el producto de su diagonal principal y muestra el resultado en el terminal.

Se pide:

- Desarrolle una solución con base en monitores teóricos que modele el sistema, utilizando variables de condición y disciplina de señalización `SC` (señalar y seguir). Guarde su solución en un documento `monitorCadena.pdf`.
- A partir del punto anterior, efectúe su implementación en lenguaje Java, encapsulando los recursos necesarios, escribiendo el conjunto de métodos que proporcionen la gestión desarrollada en los monitores teóricos, y sincronizando cuando sea necesario. Guarde el código del monitor en `monitorCadena_x.java`, siendo $x = 1, 2, ...$ para cuantos ficheros necesite al implantar la solución, y escriba ahora un programa que los utilice donde creará los hilos concurrentes correspondientes a los proceso `A`, `B` y `C`. Guarde este programa en `UsamonitorCadena.java`.

3. Escriba ahora un monitor teórico que sirva para simular a un semáforo general. Dicho monitor podrá inicializar al semáforo con el valor inicial deseado, e implementará operaciones `wait()` y `signal()` de semántica equivalente a los semáforos `Djikstra`. Guarde su solución teórica en `monitorSemaforo.pdf`. Escriba ahora un monitor en Java equivalente utilizando el API estándar, y guárdelo en monitorSemaforo.java. Finalmente, escriba un protocolo de exclusión mutua para dos procesos controlado con un objeto de la clase anterior, y compruebe que funciona como debe. Guárdelo en `mutex.java`.

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:

- Los documentos escritos que no sean ficheros de código deben generarse **obligatoriamente** utilizando Latex, a través del editor *OverLeaf*, disponible en la nube. Tiene a su disposición en el Campus Virtual un manual que le permitirá desarrollar de forma sencilla y eficiente documentos científicos de alta calidad. Puede encontar el citado manual en la sección dedicada a Latex en el bloque principal del curso virtual. El url de *OverLeaf* es: `https://www.overleaf.com/`
- 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**. utilizará paralelismo de grano grueso, particionando un problema y definiendo el número de hilos necesarios para su resolución en función de ecuaciones de balanceado de carga.

- 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.