Lectores y escritores
Práctica 3.2 - PC

Miembros del grupo:
Carlos Giraldo
Ricardo Pragnell

Ejercicio 1: lectores-escritores con monitores

- **¿Cómo funciona la notificación optimizada en su solución? (es decir, el uso de notify en vez de notifyAll en uno de los métodos del monitor). ¿Puede producirse la inanición?**

En nuestra solución hay dos puntos clave en los que despertamos a los procesos que se quedan bloqueados en un wait. En el caso de el método **salirLeer()** empleamos un **notify()** puesto que sabemos que solo se han bloqueado los escritores y basta con despertar a uno solo. En el caso de **salirEscribir()** empleamos un **notifyAll()** ya que toca despertar tanto a lectores como a escritores.

Es una solución en la que la inanición por parte de escritores y lectores podría estar muy presente dado que tanto si se da como si no todo depende de los tiempos que tienen cuando leen/escriben o descansan.

Ejercicio 2: lectores-escritores con prioridad de escritores

- **¿Cómo consigue dar prioridad a los escritores? ¿Se puede optimizar la notificación al igual que en el ejercicio 1? ¿Puede producirse la inanición?**

Empleamos un contador de escritores en espera de manera que no puedan acceder lectores al monitor si hay escritores pendtientes de escribir.

De igual manera, cuando se han marchado todos los lectores (sabiendo que solo despertaremos escritores) empleamos un **notify()** en lugar de un **notifyAll()**.

En esta solución sigue existe la posibilidad de inanición por parte de los lectores si se da el caso de que vengan muchos escritores seguidos.

Ejercicio 3: lectores-escritores sin inanición categórica

- **¿Cómo consigue la ausencia de inanición categórica? (también llamado inanición de grupo) ¿Puede producirse la inanición individual?**

Para eliminar la inanición categórica hemos empleado un sistema de turnos mediante el cual se prevee que cada vez que los lectores abandonen la base de datos le toque a un escritor acceder a la misma, y cuando este termine de escribir sea otra vez el turno de los lectores.

Virtualmente tanto lectores como escritores tienen oportunidad de acceder, pero si nos fijamos más en detalle se puede apreciar que algunos escritores pueden sufrir de inanición al no poder asegurar que se despertará cuando llegue su turno.

Ejercicio 4: lectores-escritores sin inanición individual

- **¿Cómo consigue la ausencia de inanición individual?**

Para eliminar la inanición individual hemos implementado una clase Secuenciador y hecho uso de algunas variables de control, con el fin de obtener un comportamiento FIFO.
Cada lector y escritor va pidiendo un ticket al Secuenciador.
Los lectores antes de comenzar a leer comprueban que no haya un escritor. Además para mantener la concurrencia de lectores permitimos que vayan entrando según pidan ticket y la variable **lectoresSeguidos** este a true.
Cuando un escritor entre a escribir pedirá ticket y bloqueará a todos los demás (lectores y escritores) de adquirir un ticket ya que si no el número de ticket avanza sin que llegue el turno a nadie a entrar creando un deadlock. Una vez llegue su turno escribirá y al terminar vuelve a **habilitar** la adquisición de tickets.

De esta manera eliminamos la inanición individual y mantenemos la concurrencia de lectores.