PCTR - Práctica 4

Javier López Sierra

Noviembre 2021

1. Ejercicio 1

Al ejecutar *tryThree.java* solo observamos que no se termina la ejecución por mucho tiempo que esperemos. Puedo asumir que evidentemente se trata del problema de espera infinita asociado a este algoritmo, en el cual ambos hilos se han quedado atrapados en el bucle de espera, esperando cada uno a que el otro hilo termine, sin poder terminar ninguno de los dos.

Al ejecutar la versión *tryFour.java* se comprueba que en este caso sí funciona, y da el resultado esperado, *n=0*. Es evidente que haber añadido operaciones para ocupar tiempo entre la cesión y solicitud de turno es efectivo, aunque resulte poco eficiente.

2. Ejercicio 2

Al probar el algoritmo de *Dekker* podemos comprobar que es efectivo al controlar el acceso de ambos hilos a la variable n (el resultado es n=0), aunque para el caso concreto que hemos implementado, no podemos comprobar si existe una mejora en la eficiencia con respecto al algoritmo implementado en el ejercicio anterior, por la simplicidad de las operaciones.

3. Ejercicio 3

Al implementar el algoritmo de *Eisenberg-McGuire* se ha decidido emplear solo cuatro hilos, y para probarlo, se ejecutarían sumas o restas unarias dependiendo de si su identificador es par o impar, de forma que de como resultado el valor original. Se puede comprobar así que sí funciona correctamente.

4. Ejercicio 4

Al probar el algoritmo incorrecto de *Hyman* se observa que no finaliza. Esto debe ser porque el hilo que no tiene el turno ha entrado en el bucle de espera a la sección crítica antes de que el hilo que tenía originalmente el turno pueda comprobarlo, y ha modificado la variable turno para tomar turno, forzando al

hilo que iba a tener su turno a entrar en el bucle de espera. Ambos hilos acaban atrapados en el bucle de espera porque ninguno de ellos modifica su variable C1 o C2, haciendo entender al hilo contrario que están en la sección crítica, cayendo el algoritmo en una espera infinita.