PROGRAMACION CONCURRENTE (Curso 2013/2014)

PRÁCTICA 2
FECHA REALIZACIÓN: semana 10 de marzo

Objetivos de la práctica:

1. Condición de carrera
2. Exclusión mutua

Parte 1: Provocar una condición de carrera. Escribir un programa concurrente en el que múltiples threads compartan y modifiquen una variable de tipo int de forma que el resultado final de la variable una vez que los threads terminan no sea el valor esperado. Tendremos dos tipos de procesos, decrementadores e incrementadores que realizan N decrementos e incrementos, respectivamente, sobre una misma variable (n) de tipo int inicializada a 0. El programa concurrente pondrá en marcha M procesos de cada tipo y una vez que todos los threads han terminado imprimirá el valor de la variable compartida. El valor final de la variable debería ser 0 ya que se habrán producido M $\times$ N decrementos (n–) y M $\times$ N incrementos (n++), sin embargo, si dos operaciones (tanto de decremento como de incremento) se realizan a la vez el resultado puede no ser el esperado (por ejemplo, dos incrementos podrían terminar por no incrementar la variable en 2).

Parte 2: Evitar condición de carrera con espera activa. La segunda parte consiste en evitar la condición de carrera que se produjo en el ejercicio anterior. Para ello supondremos la existencia de sólo dos procesos, que simultáneamente ejecutan sendos bucles de N pasos incrementando y decrementando, respectivamente, en cada paso una variable compartida (la operación de incremento y la de decremento sobre esa misma variable compartida son secciones críticas). El objetivo es evitar que mientras un proceso modifica la variable el otro haga lo mismo (propiedad que se denomina exclusión mutua: no puede haber dos procesos modificando simultáneamente esa variable) y el objetivo es hacerlo utilizando sólo nuevas variables y "espera activa" (en otras palabras, está prohibido utilizar métodos synchronized, semáforos o cualquier otro mecanismo de concurrencia).

Parte 3: Generalizar la parte 2 para que funcione con 2M procesos (M incremetandores y M decrementadores). Utiliza para garantizar exclusión mutua los algoritmos vistos en clase.