Computación Concurrente 2017-2. Práctica 3: Algoritmo de Filtrado para Exclusión Mutua.

José David Flores Peñaloza Susana Hahn Martín Lunas
Armando Ballinas Nangüelú

Fecha de entrega: 14 de marzo de 2017.

1. Introducción

En esta práctica se debe implementar un algoritmo de exclusión mutua para un número arbitrario de hilos para la biblioteca Pthreads.

Dicha biblioteca implementa de manera nativa objetos *mutex* que resuelven el problema de exclusión mutua. Sin embargo, el objetivo de la práctica es que los alumnos implementen su propia versión de estos objetos implementando el algoritmo de filtrado (*filter*).

El algoritmo de filtrado es una generalización del algoritmo de Peterson para un número arbitrario de hilos.

Se supone que el alumno ya realizó la implementación del algoritmo de Peterson, de modo que ya se familiarizó con la estructura de los objetos *mutex* en Pthreads, en particular con las funciones: `init`,`destroy`,`lock` y `unlock`.

2. El algoritmo de filtrado para exclusión mutua.

Este algoritmo generaliza la idea fundamental del algoritmo de Peterson para dos hilos. Si se tienen n hilos ejecutando el algoritmo, este crea $n-1$ niveles de espera para los hilos, en donde cada nivel logra lo siguiente:

- $\forall k, 1 \leq k \leq n-1$, si un hilo quiere entrar a un nivel k, entonces lo logra.
- Si más de un hilo quiere entrar a un nivel k, entonces al menos uno se queda esperando en este nivel.

El objetivo del algoritmo es crear salas de espera en distintos niveles para los hilos, de modo que el hilo que logre pasar por todos los niveles, obtendrá el candado.

Para más detalles del algoritmo consultar [Her].

3. Ejercicios.

La práctica se puede realizar en parejas y se debe enviar al correo *armandoballinas@ciencias.unam.mx* con el asunto *concurrente_p3*.

1. Realizar la implementación del algoritmo de filtrado para Pthreads implementando las funciones `init`,`destroy`,`lock` y `unlock`. Estas funciones estarán definidas en un archivo `filter.h` e implementadas en un archivo `filter.c`.

2. Realizar un programa de prueba `main.c` en donde se pruebe la implementación del candado. Este programa debe calcular el producto punto de dos vectores, lanzando un número arbitrario de hilos. Para realizar este ejercicio puede reutilizar el archivo creado para el último ejercicio de la práctica 2, cambiando el uso de objetos *mutex* nativos, por la implementación del algoritmo de filtrado.

3. En el `readme.txt` se debe discutir la comparación entre las dos versiones, es decir, se deben comparar la versión utilizando *mutex* y *filter*. Se debe establecer qué métodos de medición se utilizaron para la comparación, así como qué versión les parece mejor y por qué.

4. Referencias.

[Her] Herlihy, Maurice; Nir Shavit. *The Art of Multiprocessor Programming*, primera edición, 2008. Elsevier, páginas: 28,29,30.