Aproximaciones PC

Memoria: Práctica 4

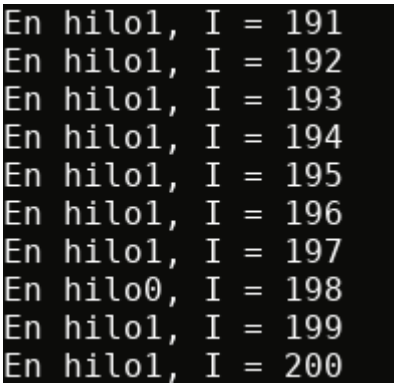
Lourdes Mas Lillo

lml26@alu.ua.es

Todas las pruebas se han realizado para distintas tallas del problema.

El algoritmo de **Dekker** lo hemos realizado con hilos java y se encuentra en el fichero Dekker.java.

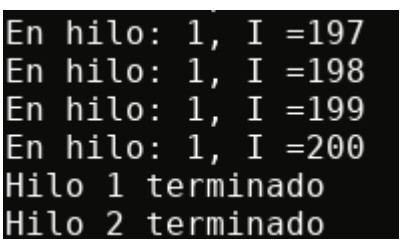
Cada hilo espera si no es su turno, mientras tanto su estado se mantendrá a falso (0), y el otro hilo que tiene el turno podrá entrar en la sección crítica mientras tanto, al terminar un hilo de utilizar la sección crítica, este cede su turno al otro hilo y de esta manera el otro puede entrar.



Dekker

Para la realización del algoritmo de **Peterson** hemos utilizado hilos POSIX, podemos encontrar el codigo en el fichero Peterson.c.

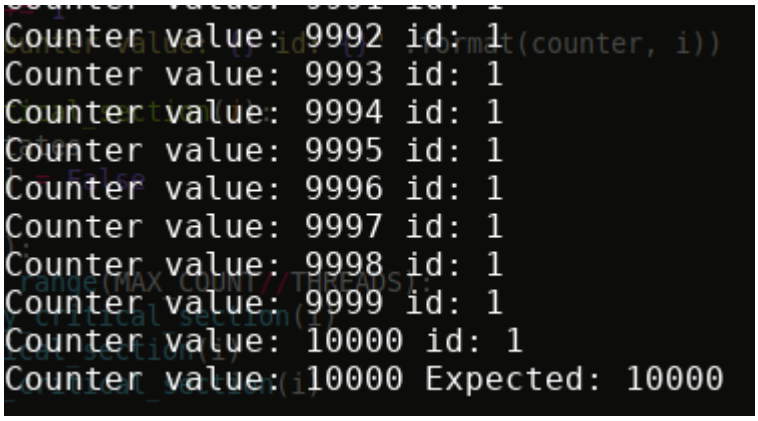
Este algoritmo es correcto y podemos comprobar en la salida para 200 iteraciones (100 para cada hilo), como la salida es correcta, esto se debe a que el algoritmo asegura la exclusión mutua, ya que cada hilo permite el paso del otro, si aquel requiere usar el uso de la sección crítica y es su turno. La espera es limitada ya que todos los hilos podrán ser ejecutados en algún momento, si no es su turno, el otro hio terminará y permitirá el paso. Es una solución similar a Decker pero más sencilla y elegante.



Peterson

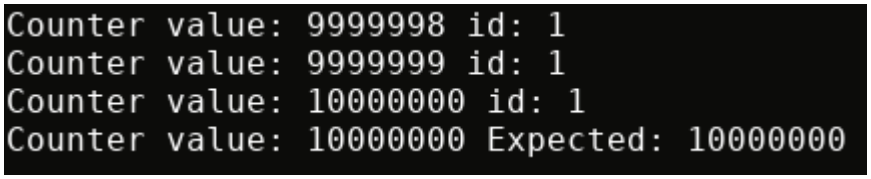
El algoritmo de **Hyman** ha sido realizado con el lenguaje Python y se encuentra en el fichero Hyman.py. Se puede dar un fallo en el acceso a la sección crítica ya que no asegura la exclusión mutua. Ambos hilos pueden acceder a la vez, si por ejemplo uno de los procesos se encuentra realizando el protocolo de acceso y justo antes de asignarse el turno, el otro proceso (o hilo en este caso) intenta acceder al protocolo de acceso.

Para 10000 iteraciones hemos obtenido un resultado correcto.



Hyman

He probado el algoritmo con otros valores más elevados para comprobar si conseguía darse el fallo, pero no ha sido así.



Hyman con 10000000 valores

El algoritmo de **Lamport** no presenta espera ilimitada debido a que a según el orden en que se solicita acceso, se le da a cada proceso un número que representa su puesto en la espera, el proceso con la numeración menor es la que entra en la sección crítica cuando esta está disponible.

El orden de acceso será en función del valor asignado, para encontrar el nuevo valor, se recoge el mayor valor hasta el momento en la cola y se le añade 1. Al utilizar esta continuidad, se puede dar un fallo al emplear un número de procesos (o hilos) mayor al máximo valor que acepta el tipo de dato empleado para dar posición en el turno.

En el algoritmo empleo una función que recorre el vector de números asignados para cada proceso, y de esta forma obtener el mayor. Sería más eficiente si en lugar de recorrer el vector cada vez, tuviera una variable global en la que se almacenara el valor más grande directamente. Aunque podría presentar problemas al haber varios procesos que quisieran leer o escribir de ella, o sea, sería una sección crítica. O también al aumentar indefinidamente, ya que los números asignados a los turnos vuelven a 0, en cuanto el proceso sale de la sección crítica.

A continuación se muestra las salidas empleando un solo procesador (taskset -c 0), para tener realmente un trabajo de coordinación entre hilos de un solo proceso, y sin la opción, donde se aprecia el intercambio de turno.

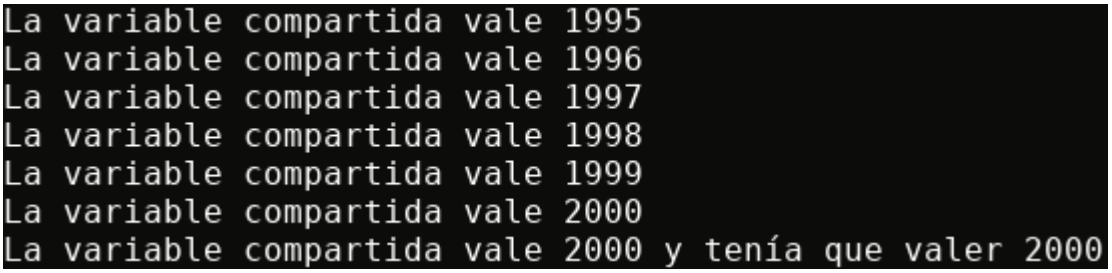
```
En hilo: 0, I =194
En hilo: 0, I =195
En hilo: 0, I =196
En hilo: 0, I =197
En hilo: 0, I =198
En hilo: 0, I =199
En hilo: 0, I =200
Hilo 0 terminado
Hilo 1 terminado
```

Lamport con opción taskset -c 0

```
En hilo: 1, I =194
En hilo: 0, I =195
En hilo: 1, I =196
En hilo: 0, I =197
En hilo: 1, I =198
En hilo: 0, I =199
En hilo: 1, I =200
Hilo 0 terminado
Hilo 1 terminado
```

Lamport no monoprocesador

El algoritmo de **espera ocupada** se ha realizado en c, y código ensamblador asm. Se encuentra en el fichero EsperaOcupada.c El resultado de la ejecución es correcto. Es un código que emplea cerrojo, al emplear exchange, el intercambio de permiso de acceso se da de manera atómica, por lo tanto no hay ningún momento ambivalente en el que varios hilos puedan acceder a la sección crítica a la vez.



Espera ocupada