Sistemas Operativos 2019-2
Tarea 4. Sincronización entre procesos

Profesor : Salvador López Mendoza
Ayudante : Jorge Erick Rivera López

Facultad de Ciencias, UNAM

Fecha de entrega: 19 de Marzo de 2019

Para esta tarea se tiene que resolver el siguiente problema visto en clase.

1. Problema

Se tiene un arreglo de 1's y 0's de longitud 6, en un tiempo t_i cada celda del arreglo definirá el valor que guardará en el tiempo t_{i+1} dependiendo del valor de las celdas adyacentes a ella en el tiempo t_i, si las celdas adyacentes tienen el mismo valor, entonces el valor de la celda que quiere actualizar su valor será 1, de lo contrario será 0, para las celdas que están en los extremos se supone que la celda que no está en el arreglo tiene siempre valor 0.

2. Instrucciones

Para resolver este problema, tendrás que modificar el código visto en clase, de tal forma que se dará una solución utilizando hilos puesto que la solución original lo resuelve de forma secuencial y sin uso de hilos, deberás convertir el código en **Java** a **C**, a diferencia del código original, el programa modificado solo funcionará devolviendo el paso del tiempo 0 al tiempo 1, no se recibirán argumentos a través de la línea de comandos, el arreglo será modificado directamente en el código fuente para su prueba.

Deberás utilizar *pthread_t* como se vió en clase, puedes utilizar las siguientes opciones para realizar tu tarea:

1. *pthread_mutex_t*
2. *sema_t*, esta estructura viene en la biblioteca *semaphore.h*

Además de esto tendrás que considerar al menos un *hilo* por celda, cada *hilo de celda* tendrá que verificar a sus vecinos, sólo se permitirá que un hilo a la vez verifique a sus *celdas vecinas*, además deberá imprimir un mensaje que indique que el hilo esta iniciando a verificar y que ha terminado de verificar.

Ayuda: Tal vez te ayude buscar una función en pthread que haga una barrera.

3. Ejemplo de entrada y salida

Suponiendo como arreglo {1,0,0,1,1,0} *

```
usuario@maquina ~]$ ./tuprograma
Comenzando a verificar hilo 0 en tiempo 0
Terminó de verificar hilo 0 en tiempo 0
Comenzando a verificar hilo 1 en tiempo 0
Terminó de verificar hilo 1 en tiempo 0
. . .
Comenzando a verificar hilo 5 en tiempo 0
Terminó de verificar hilo 5 en tiempo 0
1 0 0 0 0 0
usuario@maquina ~]$
```

Observación: La llegada de los hilos puede variar, pero debe imprimir el mismo hilo que esta verificando, su entrada y salida de la verificación de forma consecutiva como en este ejemplo.

4. Puntos extra

(1 pt) Haz que el programa funcione para n tiempos, de tal forma que en la línea de comandos se reciba como argumento el número de ciclos (Ejemplo tomando en cuenta a *, también se debe imprimir el ciclo en el que se está).

```
usuario@maquina /src ~]$ ./tuprograma 2
Comenzando a verificar hilo 0 en tiempo 0
Terminó de verificar hilo 0 en tiempo 0
. . .
Comenzando a verificar hilo 5 en tiempo 0
Terminó de verificar hilo 5 en tiempo 0
Comenzando a verificar hilo 0 en tiempo 1
Terminó de verificar hilo 0 en tiempo 1
. . .
Comenzando a verificar hilo 5 en tiempo 1
Terminó de verificar hilo 5 en tiempo 1
1 0 1 1 1 1
```

(2 pts) Utiliza **variables de condición** para resolver este problema, con lo cual no necesitarás de usar de una función barrera ya hecha, para esto investiga *pthread_cond_t*.

5. Política de retraso

Por cada semana de retraso en la entrega se restará un punto.

6. Reglas importantes

El trabajo se realizará y entregará **individualmente**, deberás entregar tu código fuente en una carpeta nombrada **src** con su respectivo **READ-ME** que contendrá únicamente tu nombre, además tu número de cuenta, y comprimido en un archivo **ZIP** que deberá llevar tu nombre (el código fuente también debe llevarlo). La entrega será a **más tardar a las 23:59**, cualquier copia detectada entre compañeros será considerada con calificación 0 en la tarea para los implicados.