Taller 0: Hilos

Malcolm Davis, Miembro Estudiantil, IEEE
mdavis.cr@ieee.org

Resumen— En este documento se encuentran las respuestas a las preguntas propuestas en el taller 0 del curso Arquitectura de Computadores II relacionadas con los Hilos de un computador.

Index Terms— TEC, Arquitectura De Computadores II, Threads, Pthreads, Mutex.

1. Pregunta 1: Métodos para Manejo de Hilos en C

Investigue posibles métodos (bibliotecas, apis, etc) para el uso de hilos bajo el sistema operativo GNU Linux. P.e. pthread. Los métodos más reconocidos tanto en C como en C++ son:

1. PTreads
2. Open MP
3. Intel Threading Building Blocks (TBB)
4. BOOST (C++)
5. POCO (C++)

Pero debido a la naturaleza del problema y a sugerencia del profesor, se utilizará pthreads en este taller.

2. Pregunta 2: Métodos para el Manejo de Memoria Compartida

¿Qué métodos existen para compartir memoria (variables globales) de manera segura entre hilos? Ya que se va a utilizar pthreads para solucionar el problema, se detallan los métodos de sincronización el manejo de memoria entre hilos.

2.1. Mutal Exclusive Objects (mutex)

Los mutex se utilizan para garantizar la atomicidad de una parte del código, en secciones criticas que deben ser ejecutadas por un hilo a la vez son útiles.

2.2. Semáforos

Los semáforos se utilizan para mantener el registro del uso de varios recursos que pueden ser utilizados o no dependiendo del estado del semáforo.

2.3. Variables Condicionales

Las variables condicionales son útiles para evitar la espera activa(un ciclo que espera activamente hasta que se cumpla otro proceso), se pueden usar para enviar señales y así sincronizar los procesos.

2.4. Barreras

Las barreras se utilizan para sincronizar que 2 o más hilos lleguen empiecen desde un punto al mismo tiempo. Se pone una barrera que detiene los hilos que están adelante hasta que la cantidad deseada llegue al punto.

3. Problema 1: Implementación de un Hilo Generador de Muestras y un Hilo que Utilice las Muestras

Diseñe una aplicación en C que contenga interacción entre 2 hilos. El primer hilo debe simular un proceso de muestreo, al generar un número entero de bits aleatorio (entre 0 y 255) y almacenarlo en un arreglo cada 10 mili segundos. El segundo hilo debe tomar datos uno a uno del arreglo, aplicar una operación XOR con el número 0X20 y mostrar el dato como un caracter (char) en la terminal.
Para este problema se propone el código que se puede encontrar en el archivo main.c que se entrega junto este documento. Algunas modificaciones para hacer más interesante la asignación se realizaron.
La primera modificación es permitir la configuración de los parámetros del programa utilizando los argumentos de linea de comandos que se le pasan al programa; Estos argumentos pueden ser insertados o obviados para correr los valores por defectos del problema.

```
$ ./run
$ ./run runTim maxSize samTim priTim nThre
```

El argumento **runTim** se utiliza para definir la cantidad de segundos en que el programa se va a ejecutar, debe ser un número entre 1 y 10. El argumento **maxSize** define la cantidad máxima de elementos que los arreglos pueden tener, debe ser un número entre 1 y 100. El argumento **samTim** define el tiempo, en micro segundos, que transcurre entre la generación de 2 muestras. El argumento **priTim** define la cantidad de micro segundos que transcurren entre la impresión en pantalla de 2 arreglos del mismo hilo. El último argumento posicional **nThre** define la cantidad de hilos que se ejecutaran, este debe de ser un múltiplo de 2 ya que crea pares generador-impresor.
Se resuelve el problema no para un par de hilos generador-impresor sino para N cantidad par de hilos. Para evitar la utilización de variables globales y aún así manejar memoria compartida entre los hilos, se maneja un arreglo de un

struct de argumentos que contienen los datos que necesitan ambos hilos conocer, y la memoria se comparte entre ellos. Además esta forma de solucionar el problema nos ayuda en el manejo de memoria para N hilos.
Aunque por la naturaleza del problema (escritura en un hilo y lectura en el otro) existen menos posibilidades de tener algún conflicto con el manejo de memoria entre 2 hilos(esto no significa que no existan problemas, un hilo podría estar escribiendo y el otro lee en ese momento y la lectura sería errónea) se utilizó la estrategia de sincronización por mutex.

REFERENCIAS

[1] Lewis, Bill and Daniel J. Berg.*Multithreaded Programming with Pthreads*. Morristown, Multithreaded Programming with Pthreads, 1998.