UNIVERSIDAD DE ANTIOQUIA

PROYECTO DE INVESTIGACIÓN: HILOS

ALUMNO

ANDREA OSPINA HINCAPIÉ
andrea.ospinah@udea.edu.co
INFORMÁTICA II
INGENIERÍA ELECTRÓNICA

DOCENTE

AUGUSTO ENRIQUE SALAZAR JIMÉNEZ
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

17 DE JULIO DE 2020

PROYECTO DE INVESTIGACIÓN: HILOS

En la actualidad es bastante común que para utilizar una aplicación, por ejemplo, sea necesario ejecutar miles y miles de procesos casi que de manera simultánea y de la forma más eficiente posible.Algunas de las herramientas utilizadas para este objetivo han sido el uso de múltiples núcleos en un procesador, de manera que sea posible ejecutar más procesos en un menor tiempo, así como el uso de un hilo o múltiples hilos asociados a un núcleo [1, p.736].

Pero, ¿qué son estos hilos?¿cómo funcionan?¿qué tipos existen? Estas y otras preguntas sobre los hilos serán respondidas a continuación:

¿Qué es un hilo en el contexto de los microprocesadores?

Un hilo puede entenderse como la ünidad básica de utilización de un procesador"[2, p.43] por lo que si se tiene, por ejemplo, una aplicación que se va a ejecutar, entonces se crea un proceso pesado al que se le asignan recursos y que, a su vez, se divide en procesos ligeros (hilos) [3, p.68], de manera que puedan entenderse como ramas del programa que se ejecutan en paralelo. [4, p.70]. Cada hilo tiene su propio conjunto de registros, contador de programa y pila [2][3] que corresponden a su ID, mientras que comparten recursos asignados al proceso duro con los otros hilos, como las señales de un sistema operativo o los archivos que se encuentren abiertos [2]. Es por medio de hilos que, entonces, se le da solución a diversos problemas como el tener una base de dato con múltiples usuarios realizando peticiones de acceso a esta de manera simúltanea. Sin el uso de múltiples hilos solo se podría permitir el acceso a un usuario a la vez.

¿Se puede hablar de la historia de los hilos?

El concepto de hilos y uso de múltiples hilos ha existido desde la mitad del siglo pasado. El computador DYSEAC, el primer computador portable, comenzó su funcionamiento en 1954 y fue uno de los primeros en utilizar múltiples hilos e incluir dos contadores de programa. Durante el mismo periodo de tiempo en el que fue desarrollado el anterior computador se creó también el LINCOLN TX-2 en los laboratorios del MIT, el cual, con el objetivo de manejar múltiples dispositivos I/O según un nivel de prioridad, implementó múltiples hilos y hasta 33 contadores de programa [5, p.83]. En los siguientes años, mútliples laboratorios continuaron investigando el uso de procesadores que utilizan múltiples hilos hasta que en el año 2000 salió al mercado el procesador de Intel Pentium 4, el cual se convirtió en el primer procesador comercial y de propósito general en usar múltiples hilos o, como fue patentado por Intel, el primero en usar "hyper-threading"[5, p.91]. Por otro lado, puede ser interesante también conocer las soluciones a los problemas que los hilos solucionan actualmente antes de que estos se popularizaran. Un ejemplo de estas era la creación de procesos para responder a múltiples peticiones simultáneas en lugar de crear múltiples hilos. Esta solución no era eficiente ya que demandaba el uso de abundantes recursos de procesamiento para la creación de múltiples procesos [2, p.43].

¿Que tipo de hilos existen?

Existen dos tipos distintos de hilos así como combinaciones de estos tipos, ofrecidas por sistemas operativos específicos como es el caso de Solaris,un sistema operativo de Unix [6, p.192]. Los tipos de hilos son:

Hilos a nivel de usuario(ULT):

Son aquellos que se gestionan sin soporte de un sistema operativo y existen en tiempo de ejecución. Estos son implementados por medio de una librería cuyo código se encuentra en el espacio de usuario y por lo tanto este dispone del código de manera similar a como dispone de módulos propios desarrollados, permitiendo el uso de algoritmos personalizados para planificar los hilos de un proceso. [3][7].

A diferencia de los hilos a nivel de núcleo, los hilos a nivel de usuario utilizan el planificador del runtime por lo que el núcleo no identifica la existencia de múltiples hilos de usuario y los trata como uno solo, esto permite que los hilos puedan ser implementados así el sistema operativo que se esté usando no soporte hilos [8].

Hilos a nivel de núcleo (KLT):

Son aquellos manejados por el sistema operativo a través de llamadas al sistema [3] de manera que el sistema interrumpe la ejecución de un hilo para ejecutar otro según un orden establecido por el sistema operativo. Esto hace que el cambio entre hilos a nivel de núcleo sea más lento comparado con el cambio entre hilos a nivel de usuario [8, p.401].

¿Cómo se hace la implementación de hilos a nivel de hardware?

La implementación de hilos a nivel de hardware depende en gran medida de si se utiliza una arquitectura con un único o múltiples procesadores o núcleos. En el caso en que solo se tiene un procesador los hilos son implementados cambiando rápidamente entre hilos de software para crear una ilusión de paralelismo ya que no es posible ejecutar varios hilos simultáneamente. Por lo tanto, y como es de esperarse, en un procesador con múltiples núcleos o en un sistema con múltiples procesadores sí es posible tener un verdadero paralelismo entre hilos al cada procesador o núcleo ejecutar por lo menos un hilo. Este último proceso ha recibido distintos nombres según la empresa fabricante de procesadores con múltiples núcleos. En el caso de Intel el uso de múltiples hilos recibe el nombre de ”hyper-threading”, mientras que AMD lo denomina ”simultaneous multithreading”. [7, p.402].

¿Cómo se implementan los hilos por software?

La implementación de hilos a través de software depende tanto del lenguaje de programación utilizado como del sistema operativo en el que se realiza. En el caso de los sistemas operativos una de las diferencias fundamentales entre implementaciones de hilos es el número de hilos usados por proceso. En el caso de implementaciones tradicionales de Unix lo común es que por cada proceso solo exista un hilo, y para otros sistemas operativos como Linux, Solaris y Windows se pueden tener múltiples hilos por proceso [6, p.169]. Otra diferencia radica en la relación entre hilos a nivel de usuario e hilos a nivel de núcleo, lo cual establece el tipo de modelo multihilo que se esté usando. Windows y Linux utilizan un modelo uno a uno en el que a un hilo de usuario se le asigna un hilo de núcleo que permite la ejecución de múltiples hilos simultáneamente sobre varios procesadores. Sin embargo, la creación de hilos de núcleo puede afectar el rendimiento de la aplicación por lo que este tipo de implementaciones suelen limitar el número de hilos que puede soportar el sistema [2, p.45]. Para que el programar pueda implementar los hilos es común el uso de librerías de hilos que permite tener una API(application programming interface) para la gestión de estos. El uso de estas librerías puede darse ya sea en el espacio de usuario o en el espacio del kernel, involucrando esto último llamadas al kernel por lo que la implementación de estas depende, de nuevo, del sistema operativo

[2, p.46]. En el caso de los lenguajes de programación puede encontrarse una diferencia en el grado de dificultad involucrado en la creación de hilos. Java, por ejemplo, fue creado como un lenguaje para implementar múltiples hilos por lo que sus objetos contienen un candado que facilita la comunicación y sincronización entre hilos, así como palabras reservadas que facilitan esta sincronización. Lo anterior hace que la implementación de hilos en Java implique menos trabajo que, por ejemplo, en C++, un lenguaje que para lanzar y sicronizar múltiples hilos es necesario utilizar una o múltiples librerías como la librería threads (a partir de C++11) [9, p.126].

Implementación de hilos

A continuación se mostrará una implementación de hilos en C++ para lo cual debe tenerse en cuenta algunos de los siguientes elementos:

Hilos en C++

-Existen dos maneras de implementar hilos en C++: la primera consiste en el uso de la librería pthread, incluída dentro del POSIX standard, usado por sistemas operativos como Linux y Mac, pero no directamente por Windows. La segunda manera es con el uso de la librería threads a partir de C++11, la cual es la forma nativa de implementación al hacer parte de C++ estándar [10, p.284]. Por las ventajas que ofrece la librería threads se usará esta en el ejemplo de implementación.

-Para la implementación de hilos se crea por lo menos una función a ejecutar por los hilos. Usando std::thread esta función puede recibir tantos parámetros como se desee [11, p.748].

-Para lanzar un hilo se utiliza la funciń std::thread con un primer parámetro que será el nombre de la función a ejecutar y los siguientes parámetros serán los valores de los parámetros de la función que se ejecutará (Esto se implementa usando punteros a funciones)[11, p.748].

-Al lanzar los hilos es necesario llamar join() en ejemplos sencillos. Lo que este join() permite hacer es que se continúe ejecutando el main al lanzar los hilos [11, p.748].

-Cuando se utilizan hilos se debe tener cuidado cuando múltiples hilos comparten información.Para evitar problemas deben sincronizarse los hilos. Existen múltiples formas de realizar esta sincronizaciones dependiendo de la complejidad de los datos, como usar las funciones de la librería atomic [11, p.758] o la sincronización explícita usando la clase mutex que implementa métodos como lock() y unlock() para el manejo de información [11, p.759].

Ejemplo de implementación

El siguiente ejemplo utiliza múltiples hilos para ejecutar dos funciones: la primera función se encarga de duplicar un número que se le pasa por referencia y la segunda función se encarga de sumar dos números que se le pasan como parámetros. Además, los hilos que ejecuten la función 1 deberán dormir durante 10 segundos después de realizar la multiplicación, mientras que el hilo que ejecuta la función 2 deberá dormir por 3 segundos antes de realizar la suma. Por último, se mostrarán los tiempos de inicio y finalización de la ejecución de cada función. Para esto se utilizará la librería threads, chrono, ctime y mutex para lanzar los hilos, obtener los tiempos de ejecución y sincronizar los hilos.

El código fuente de la implementación puede encontrarse en esta carpeta del repositorio, junto con el .pro utilizado para su ejecución correcta en QtCreator. Este código fue implementado en Ubuntu 18.04 LTS.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. Chivers, *Threads*. London: Springer London, 2003, pp. 169–183. [Online]. Available: https://doi.org/10.1007/978-1-4471-0075-1_11

[2] L. D. Rodríguez, *El gran libro del PC interno*. Marcombo, 2007.

[3] S. Candela Solá, C. GARCÍA RUBEN, A. QUESADA ARENCIBIA, F. J. SANTANA PÉREZ, and J. M. SANTOS ESPINO, *Fundamentos de sistemas operativos. Teoría y ejercicios resueltos: teoría y ejercicios resueltos*. Editorial Paraninfo, 2007.

[4] J. M. M. Pascual and J. A. P.-C. Atanasio, *Conceptos de sistemas operativos*. Univ Pontifica Comillas, 2002, vol. 4.

[5] M. Nemirovsky and D. M. Tullsen, "Multithreading architecture," *Synthesis Lectures on Computer Architecture*, vol. 8, no. 1, pp. 1–109, 2013.

[6] W. Stallings, *Operating systems: internals and design principles*. Upper Saddle River, NJ: Pearson/Prentice Hall„ 2009.

[7] K. Shibu, *Introduction to embedded systems*. Tata McGraw-Hill Education, 2009.

[8] M. Silva, *Sistemas operativos*. Alfaomega Grupo Editor, 2015.

[9] K. D. Lee, *Foundations of programming languages*. Springer, 2017.

[10] J. Swaminathan, M. Posch, and J. Galowicz, *Expert C++ Programming*. Packt Publishing, 2018.

[11] M. Gregoire, *Professional C++*. John Wiley & Sons, 2014.