LOS HILOS A NIVEL DEL MICROPROCESADOR

Liliana Marcela Barbosa Esteban
Universidad de Antioquia
Curso: Informática II

1. INTRODUCCIÓN

En los últimos años, el desarrollo y la evolución de los aparatos electrónicos a nuestro alrededor ha sido de manera vertiginosa, tanto es así que han abarcado gran parte de los aspectos de nuestro día. Esto ha sido posible, gracias al desarrollo de la electrónica digital, la cual permitió la aparición de los microprocesadores usados actualmente.

El microprocesador es un circuito integrado de gran eficiencia que se encuentra en todo dispositivo electrónico y actúa como la unidad central de procesamiento de la máquina. "Su rendimiento se determina por varios factores: la frecuencia de reloj, el número de bits que utiliza, la memoria caché y la cantidad de núcleos."[9]. Estos núcleos o *cores* son como subprocesadores que dentro del microprocesador le permiten realizar varias tareas al mismo tiempo.

Por otro lado, los hilos son una especie de versión virtual de los núcleos. Son los que permiten definir el flujo de control de un programa dándole la ilusión al usuario de que la máquina puede hacer más de una cosa al mismo tiempo. Esto lo logra mediante la división de las tareas y la alternación de ellas.

Puesto que durante los últimos años los microprocesadores forman parte de la mayoría de los elementos con los que se interactúa día a día, es importante estudiar a fondo los hilos ya que son los elementos que aprovechan con creciente productividad los tiempos de espera entre procesos, optimizando el funcionamiento del procesador.

Para este trabajo se toma de referencia un conjunto de documentos y páginas web con información verídica, donde los autores explican desde diferentes puntos qué son los hilos a nivel del microprocesador, y su implementación por *software* y *hardware*. Por otra parte, cabe resaltar que el tema se aborda de manera expositiva, retomando una parte de la historia de ese método y concluyendo en la importancia de él.

2. CUERPO DEL TRABAJO

Un proceso es una serie de recursos asociados con un cálculo en una zona específica de memoria. Básicamente, es un programa en ejecución que puede tener uno o varios

hilos. Es importante mencionar que un hilo es un contexto de ejecución, dónde se encuentra la información necesaria para que el núcleo del microprocesador pueda ejecutar una secuencia de instrucciones. Además, los *threads* usan la misma zona de memoria pero diferente posición en el *stack* y los registros, lo que conlleva a que si alguno de ellos altera una variable los demás hilos verán el cambio (lo que no ocurriría con los procesos).

Algunas de las variadas ventajas de los hilos son: "capacidad de respuesta, cambio de contexto más rápido, utilización efectiva del sistema multiprocesador, compartir recursos, mejor comunicación y rendimiento mejorado del sistema."[6].

A pesar de que los hilos son un concepto bastante ligado a los microprocesadores, "la noción de flujo secuencial de control, con la que se conoce actualmente, se remonta a 1965 con el sistema Berkeley Timesharing."[11]. Lo anterior implica que el concepto nació antes del microprocesador en sí, llamándose proceso y no hilo. Por otra parte, la primera implementación de aquel concepto en un sistema operativo fue en 1970, Max Smith fue el encargado de la implementación en *Multics (Multiplexed Information and Computing Service).*

No obstante, no fue sino hasta los años 80 cuando los sistemas con un solo *core* en el microprocesador, lograron brindar la ilusión de paralelismo mediante la distribución de recursos de procesamiento limitados entre procesos, a través del concepto de hilo de una secuencia de instrucciones con direcciones virtuales. Posteriormente, a inicios del 2000, "debido a problemas de consumo de potencia y temperatura en la industria se empezaron a diseñar procesadores *multicore*. En los que a su vez cada uno de estos núcleos era capaz de dar soporte a varios hilos de ejecución independientes."[12].

Cabe mencionar que actualmente existen dos tipos de *threads*, los cuales son: "los hilos a nivel de usuario y a nivel del *kernel*."[10]. Es importante resaltar que el *kernel* es una parte fundamental del sistema operativo que gestiona los recursos de la máquina haciéndole llamados al sistema. Continuando con los diferentes hilos, los primeros son implementados por los programadores sin el soporte del *kernel* y los otros son los implementados por el sistema operativo los que les permiten a los núcleos realizar múltiples tareas.

En el momento de la implementación, "el programador debe asignarle subprocesos al núcleo usando alguna de las siguientes estrategias: *Many-To-One Model, One-To-One Model, Many-To-Many Model.*"[4]. El primero de ellos, el modelo de muchos a uno *(Many-To-One Model)*, es una estrategia manejada por la biblioteca de hilos del usuario y consiste en la asignación de muchos hilos a uno solo a nivel del *kernel*. Este modelo es notablemente eficiente, a excepción del hecho de la paralización completa de la ejecución al hacer un llamado de bloqueo; lo que ocurre a causa de que un solo hilo del *kernel* sólo puede operar en un *core*. Luego, en el modelo uno a uno *(One-To-One Model)*, se crea un hilo a nivel del *kernel* cada vez que se genere uno a nivel de usuario. Aunque ello permite superar el inconveniente en el primer modelo, trae consigo problemas de ralentización del sistema como consecuencia de la sobrecarga. Por último, respecto al modelo de muchos a muchos *(Many-To-Many Model)*, se puede resaltar que es una estrategia muy eficiente ya que supera los problemas de los modelos anterior-

mente mencionados, donde la cantidad de hilos a nivel de usuario se le asigna al número igual o menor de hilos a nivel del *kernel*.

Cuando la implementación se trata a nivel de usuario, el *kernel* no sabe nada respecto a los hilos y procede a tratarlos como si fueran un simple hilo sin hacer diferencia en el tipo de método que se trata. Por otra parte, "el *kernel* depara una agrupación de hilos a la aplicación en proceso para que ella distribuya los hilos a ejecutar en cada uno, pero el trabajo de la ejecución queda al control del mismo *kernel*."[13].

En particular, al momento de la implementación por *software* de los hilos debemos considera el lenguaje y el *hardware* en el que se va a trabajar. Esto es debido a que, por ejemplo, en lenguajes como C/C++ necesitamos usar bibliotecas especificas que dependen del sistema operativo en el que se vaya a trabajar, mientras que lenguajes como Delphi o Java, están diseñados con el fin de permitirle al programador el control directo con los hilos sin necesidad de diversas librerías. Además, existen microprocesadores y sistemas operativos que requieren de una cierta manera de interacción de los hilos; "un ejemplo es Sparc/Solaris. El sistema operativo pone en marcha uno de los *threads* y no le quita el control hasta que él lo determine. Los demás *threads* quedan parados hasta que el primero «ceda el control»."[7].

3. CONCLUSIONES

Retomando las funcionalidades de los hilos a nivel del microprocesador, se puede resaltar que son un método que definen el flujo de control de un programa, que a su vez brindar la ilusión de que la máquina puede realizar muchas tareas al tiempo. La apariencia que se puede observar de multitareas se da gracias a la alternación de instrucciones de manera ágil con ayuda de los hilos.

Aquel control de la eficiencia de la máquina puede quedar en manos del programador mediante la implementación de los hilos a nivel de usuario, lo cual en el momento del "desarrollo es de gran importancia ya que a través de los hilos se puede lograr una aplicación que funcione con gran eficiencia a nivel de consumo de recursos de la máquina."[14].

Referencias

[1] «El Grupo Informático,» 7 Noviembre 2017. [En línea]. Available: https://www.elgrupoinformatico.com/que-son-los-nucleos-hilos-procesador-t39601.html.

[2] B. Voigt, «Stackoverflow,» 5 Febrero 2011. [En línea]. Available: https://stackoverflow.com/questions/5201852/what-is-a-thread-really.

[3] W. «Stackoverflow,» 20 Agosto 2017. [En línea]. Available: https://stackoverflow.com/questions/5593328/software-threads-vs-hardware-threads

[4] A. Silberschatz, G. Gagne y P. Baer Galvin, Operating System Concepts, Ninth Edition, Wiley.

[5] H. Ryckeboer, N. Casas, G. De Luca, M. Cortina, G. Puyo y W. Valiente, «La implementación de estructuras de hilos de usuario en un sistema operativo didáctico».

[6] A. Dusey, «GreeksforGreeks,» [En línea]. Available: https://www.geeksforgeeks.org/threads-and-its-types-in-operating-system/.

[7] «Ejemplos java y C/linux,» 4 Febrero 2007. [En línea]. Available: http://www.chuidiang.org.

[8] «A Brief History of Threads,» de Multithreading in the Solaris™ Operating Environment, 2002, p. 49.

[9] A. González González, «MICROPROCESADOR,» Tejina, 2020.

[10] «GreeksforGreeks,» [En línea]. Available: https://www.geeksforgeeks.org/thread-in-operating-system/.

[11] B. O'Sullivan, «teideal glic deisbhéalach,» 9 Junio 2005. [En línea]. Available: http://www.serpentine.com/blog/threads-faq/the-history-of-threads/.

[12] C. Gómez, M. Serrano, M. Gómez y J. Sahuquillo, «Una Nueva Metodología para el Estudio de Procesadores Realistas en las Titulaciones de Informática».

[13] «Operating systems,» [En línea]. Available: http://www.it.uu.se/education/course/homepage/os/vt18/module-4/implementing-threads/.

[14] E. Rodríguez García, «Núcleos e hilos en un procesador: qué son y en qué se diferencian,» Omicrono - El Español, 2017.