Proyecto investigacion-hilos.

Jose Daniel Rivera

Julio 2020

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

Un hilo es una unidad básica de utilización de CPU, la cual contiene un id de hilo, su propio program counter, un conjunto de registros, y una pila; que se representa a nivel del sistema operativo con una estructura llamada TCB (thread control block). Los hilos comparten con otros hilos que pertenecen al mismo proceso la sección de código, la sección de datos, entre otras cosas. Si un proceso tiene múltiples hilos, puede realizar más de una tarea a la vez (esto es real cuando se posee más de un CPU).

¿**Se puede hablar de la historia de los hilos?**

Hasta la década de 2000, la mayoría de las computadoras de escritorio tenían sólo una CPU de un solo núcleo, sin soporte para hilos de hardware, aunque las discusiones todavía se utilizan en dichos equipos, porque el cambio entre las discusiones fue en general sigue siendo más rápido que el proceso completo- cambios de contexto. En 2002, Intel añadió soporte para multihilo simultáneo a la Pentium 4 procesador, bajo el nombre hyper-threading ; en 2005, se presentó el de doble núcleo Pentium D procesador y AMD introdujeron el de doble núcleo X2 Athlon 64 procesador.[Costa-Castelló et al., 2010]

.

¿Que tipo de hilos existen?

los hilos pueden ser implementados a nivel de usuario o a nivel de kernel:

Hilos a nivel e usuario: son implementados en alguna librería. Estos hilos se gestionan sin soporte del SO, el cual solo reconoce un hilo de ejecución. Hilos a nivel de kernel: el SO es quien crea, planifica y gestiona los hilos. Se reconocen tantos hilos como se hayan creado. [Llaven, 2015]

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

La implementación de los hilos a nivel de hardware se derivan de las implicaciones de rendimiento y se pueden explicar de una forma en como los hilos aumentan la eficiencia de la comunicación entre programas en ejecución. En la mayoría de los sistemas en la comunicación entre procesos debe intervenir el núcleo para ofrecer protección de los recursos y realizar la comunicación misma. En cambio, entre hilos pueden comunicarse entre si sin la invocación al núcleo. Por lo tanto, si hay una aplicación que debe implementarse como un conjunto de unidades de ejecución relacionadas, es más eficiente hacerlo con una colección de hilos que con una colección de procesos separados.

¿Cómo se implementan los hilos por software? Debe quedar claro si el lenguaje de programación importa y si el hardware usado afecta. .

La programación de hilos es una de las cosas que ha caracterizado a los lenguajes Orientados a Objetos como Java, C, etc. Sin embargo eso no significa que los hilos solo puedan utilizarse en Lenguajes Orientados a Objetos también se puede utiliza en cualquier otro lenguaje. los hilos comparten el mismo espacio de direcciones y otros recursos como pueden ser archivos abiertos. Cualquier modificación de un recurso desde un hilo afecta al entorno del resto de los hilos del mismo proceso. Por lo tanto, es necesario sincronizar la actividad de los distintos hilos para que no interfieran unos con otros o corrompan estructuras de datos..[Sierra, 2007]

Mostrar un ejemplo de implementación de hilos usando lenguaje C++.

include¡iostream¿
include¡vector¿
include¡thread¿

void f(std::vector¡int¿ v);

int main()
std::vector¡int¿even = 0, 2, 4, 6, 8;
std::vector¡int¿odd = 1, 3, 5, 7, 9;
/* Lanza dos threads */
std::thread t1std::bind(f, even);
std::thread t2std::bind(f, odd);
/* Espera que finalicen */
t1.join();
t2.join();
void f(std::vector¡int¿ v)
/* Este mutex será compartido por todas las llamadas * a f() y servirá para atomizar la salida a stdout */
static std::mutex m;
/* Imprime cada elemento */
for (std::vector¡int¿::iterator it = v.begin();
it != v.end();
++it)
m.lock();
std::cout « *it “ std::endl;
m.unlock();

El siguiente programa crea dos hilos de ejecución, los objetos t1 y t2 del tipo std::thread, y pasa un puntero a función con argumentos utilizando la función std::bind. Cada llamada a función se ejecutará en un hilo de ejecución separado. Aunque la función es la misma, cada llamada recibe uno de los dos vectores even y odd. Dentro de f() hay un std::mutex declado static, por lo que todas las llamadas a f() comparten el mismo objeto m y esto permite sincronizar la escritura a salida estándar a través de las distintas llamadas. Luego de crear los objetos, la función main() espera la finalización de cada hilo llamando al método join(). El constructor de la clase std::thread lanza el hilo sin necesidad de más interacción desde la función que lo crea. Finalmente la función f() itera el vector e imprime un elemento por línea, a salida estándar.
»”

»"Referencias

[Costa-Castelló et al., 2010] »"Costa-Castelló, R., Vallés, M., Jiménez, L. M., Díaz-Guerra, L., Valera, A., and Puerto, R. (2010). Integración de dispositivos físicos en un laboratorio remoto de control mediante diferentes plataformas: Labview, matlab y c/c++. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 7(1):23–34.

[Llaven, 2015] »"Llaven, D. S. (2015). *Sistemas Operativos: Panorama para ingeniería en computación e informática*. Grupo Editorial Patria.

[Sierra, 2007] »"Sierra, F. J. C. (2007). *Programación orientada a objetos con C++*, volume 3. Grupo Editorial RA-MA.