HILOS EN LOS MICROPROCESADORES.

DIEGO FERNANDO DÍAZ TORRES*

15 de julio de 2020

¿Qué es un hilo?

Un hilo es una unidad básica de utilización de CPU, la cual contiene un id de hilo, su propio programa counter (contador), un conjunto de registros, y una pila; que se representa a nivel del sistema operativo con una estructura llamada TCB (thread control block) [6].
También se puede decir que un hilo es el encargado de administrar las tareas de un procesador y de sus diferentes núcleos, de tal manera que su desempeño sea el más eficiente. Debido a los hilos, las tareas o procesos de un programa, pueden ser divididas en pequeños trozos, de tal forma que se puedan optimizar y reducir los tiempos de espera de cada instrucción en la cola de un proceso [3]. 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).
Normalmente en los procesadores que están compuestos por 6 núcleos y 12 hilos, son capaces de dividir las tareas a realizar en 12 tareas diferentes, en vez de realizar solo 6, de esta forma es como se optimizan los tiempos de ejecución de un proceso.

Historia de los hilos

El término de "hilos" apareció por primera vez bajo el término de "tareas" en OS / 360 multiprogramación con un número variable de Tareas (MVT) en el año 1967 [2].

Tipos de hilos

1. **Hilos del núcleo (Kernel threads):** Son hilos asociados al código del núcleo del sistema operativo, creados y gestionados por el propio núcleo.

2. **Hilos de usuario (user threads):** Son parte del código de un determinado proceso de usuario. Estos a su vez se pueden dividir en proceso monohilo (programa que consta de un único hilo de usuario) y proceso multihilo (programa que se descompone en varios hilos de usuario). Es importante resaltar que para la gestión de los hilos de usuario el programador utiliza una librería de hilos, la cual, contiene funciones para la creación, destrucción, planificación y sincronización de los hilos de usuario.

*INGENIERIA ELECTRONICA, UNIVERSIDAD DE ANTIOQUIA, 2020

3. **Procesos ligeros (lightweight processes):** También denominados procesadores virtuales. Son hilos de usuario cuya gestión es realizada por el núcleo del sistema operativo. Un proceso ligero puede estar asociado a uno o varios hilos de usuario y tiene que tener asociado un hilo de núcleo [4].

Hilos a nivel de Hardware

En una aplicación KLT (Hilos a nivel de núcleo) pura, todo el trabajo de gestión de hilos lo realiza el kernel. En el área de la aplicación no hay código de gestión de hilos, únicamente un API (interfaz de programas de aplicación) para la gestión de hilos en el núcleo. Windows 2000, Linux y OS/2 utilizan este método. Linux utiliza un método muy particular en que no hace diferencia entre procesos e hilos, para linux si varios procesos creados con la llamada al sistema "clone" comparten el mismo espacio de direcciones virtuales el sistema operativo los trata como hilos y lógicamente son manejados por el kernel [5].
Algunas de las ventajas que esto brinda son las siguientes:

1. El kernel puede planificar simultáneamente múltiples hilos del mismo proceso en múltiples procesadores.
2. Si se bloquea un hilo, puede planificar otro del mismo proceso.
3. Las propias funciones del kernel pueden ser multihilo.

Hilos a nivel de Software

En una aplicación ULT (Hilos a nivel de usuario) pura, todo el trabajo de gestión de hilos lo realiza la aplicación y el núcleo o kernel no es consciente de la existencia de hilos. Es posible programar una aplicación como multihilo mediante una biblioteca de hilos. La misma contiene el código para crear y destruir hilos, intercambiar mensajes y datos entre hilos, para planificar la ejecución de hilos y para salvar y restaurar el contexto de los hilos. Cabe resaltar que el lenguaje de programación si importa, debido a que no todos soportan la implementación por hilos. En el caso de los lenguajes de programación interpretados, solo algunos pueden implementarse hilos (por ejemplo, Rubí MRI para Ruby, CPython para Python).
Todas las operaciones descritas se llevan a cabo en el espacio de usuario de un mismo proceso. El kernel continúa planificando el proceso como una unidad y asignándole un único estado (Listo, bloqueado, etc.) [5].
Algunas de las ventajas que brinda son:

1. El intercambio de los hilos no necesita los privilegios del modo kernel, porque todas las estructuras de datos están en el espacio de direcciones de usuario de un mismo proceso. Por lo tanto, el proceso no debe cambiar a modo kernel para gestionar hilos. Se evita la sobrecarga de cambio de modo y con esto el sobrecoste u overhead.
2. Se puede realizar una planificación específica. Dependiendo de que aplicación sea, se puede decidir por una u otra planificación según sus ventajas.

Ejemplo

A continuación, se muestra un segmento de código en el lenguaje de programación C++ que mediante el uso de hilos recibe como parámetro de entrada un número, lo incrementa en una unidas y posteriormente lo devuelve mediante el uso de otro hilo, el cual tiene como objetivo esperar el resultado y posteriormente mostrarlo [1].

```
//Se incluyen cada una de las librerias necesarias
#include <stdio.h>
#include <stdlib.h>
//La Libreria pthread.h es la que permite la creacion y eliminacion de los hilos.
#include <pthread.h>
#include <unistd.h>
//Esta funcion es la encargada de ejecutar cada uno de los hilos del programa
//En esta funcion se recibe un numero y se hace el incremento en una unidad.
void* thread_run(void* data)
{ sleep(2);
  printf("[TH_1:%ld]: Hello from the thread \n", pthread_self());
  sleep(1);
  (*(int*)data)++;
  printf("[TH_1: %ld]: To exit ..............\n",pthread_self());
  pthread_exit(data);}
//Cuerpo principal del programa
int main()
{
//Se realiza la creacion del hilo mediante pthread_create
  pthread_t thread;
  int data=0;
  int thread_rc;
  printf("[MAIN:%ld]: Starting ..... \n",pthread_self());
//Se verifica que el hilo no se haya creado con anterioridad
if ((thread_rc=pthread_create(&thread,NULL,thread_run,&data))!=0)
  {printf("Error creating the thread. Code %i",thread_rc);
   return -1;}
//Retardo de tiempo
sleep(1);
  printf("[MAIN:%ld]: Thread allocated \n",pthread_self());
  int *ptr_output_data;
//Mediante pthread_join se recoge el puntero que el hilo devuelve.
  pthread_join(thread,(void **)&ptr_output_data);
  printf("[MAIN:%ld]: Thread returns %d \n",pthread_self(), *ptr_output_data);
  return 0;
}
```

Referencias

[1] 11.2. Hilos. http://www.it.uc3m.es/pbasanta/asng/course_notes/c_threads_functions_es.html. (document)

[2] Hilo (informática) - Thread (computing) - qwe.wiki. https://es.qwe.wiki/wiki/Thread_(computing). (document)

[3] Qué son los hilos de un procesador? Diferencias con los núcleos. https://www.profesionalreview.com/2019/04/03/que-son-los-hilos-de-un-procesador/. (document)

[4] SOBUNIX: implementación y control de procesos multihilos – aprende y programa. https://aprendeyprogramablog.wordpress.com/2017/11/07/sobunix-implementacion-y-control-de-procesos-multihilos/. 3

[5] Usos mas comunes de los hilos | sistemasoper2. https://sistemasoper2.wordpress.com/2014/10/21/usos-mas-comunes-de-los-hilos/. (document)

[6] ¿Que son los hilos de un procesador?. https://www.fing.edu.uy/tecnoinf/mvd/cursos/so/material/teo/so05-hilos.pdf. (document)