5. Hilos

En el capítulo anterior, se suponía que un proceso consistía únicamente en un hilo. Ahora, la mayoría de los sistemas operativos proporcionan características que permiten que un proceso tenga múltiples hilos de control. En este capítulo veremos que es un hilo, sus ventajas, y los diferentes modelos de implementación.

1. ¿Qué es un hilo?

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).

Veamos un ejemplo para clarificar el concepto:
Un servidor web acepta solicitudes de los clientes que piden páginas web. Si este servidor tiene varios clientes y funcionara con un solo hilo de ejecución, solo podría dar servicio a un cliente por vez, y el tiempo que podría esperar un cliente para ser atendido podría ser muy grande.
Una posible solución sería que el servidor funcione de tal manera que acepte una solicitud por vez, y que cuando reciba otra solicitud, cree otro proceso para dar servicio a la nueva solicitud. Pero crear un proceso lleva tiempo y utiliza muchos recursos, entonces, si cada proceso realizará las mismas tareas ¿Por qué no utilizar hilos? Generalmente es más eficiente usar un proceso que utilice múltiples hilos (un hilo para escuchar las solicitudes, y cuando llega una solicitud, el lugar de crear otro proceso, se crea otro hilo para procesar la solicitud)

2. Ventajas de usar hilos

- Respuesta: el tiempo de respuesta mejora, ya que el programa puede continuar ejecutándose, aunque parte de él esté bloqueado.
- Compartir recursos: los hilos comparten la memoria y los recursos del proceso al que pertenecen, por lo que se puede tener varios hilos de ejecución dentro del mismo espacio de direcciones.
- Economía: Es más fácil la creación, cambio de contexto y gestión de hilos que de procesos.
- Utilización múltiples CPUs: permite que hilos de un mismo proceso ejecuten en diferentes CPUs a la vez. En un proceso mono-hilo, un proceso ejecuta en una única CPU, independientemente de cuantas tenga disponibles.

3. Hilos a nivel de usuario y de kernel

Hasta ahora hemos hablado de los hilos en sentido genérico, pero a nivel práctico 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.

Los hilos a nivel de usuario tienen como beneficio que su cambio de contexto es más sencillo que el cambio de contexto entre hilos de kernel. A demás, se pueden implementar aún si el SO no utiliza hilos a nivel de kernel. Otro de los beneficios consiste en poder planificar diferente a la estrategia del SO.

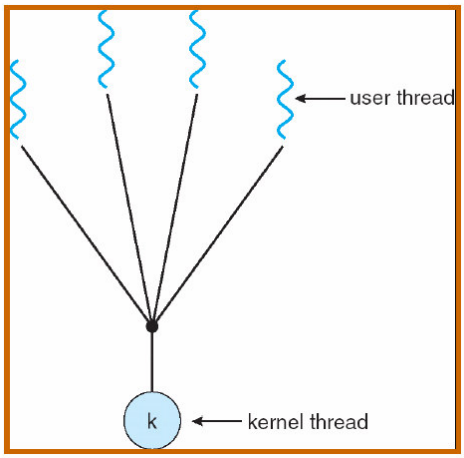
Los hilos a nivel de kernel tienen como gran beneficio poder aprovechar mejor las arquitecturas multiprocesadores, y que proporcionan un mejor tiempo de respuesta, ya que si un hilo se bloquea, los otros pueden seguir ejecutando.

4. **¿Cómo se relacionan los hilos a nivel de kernel y los de usuario?**

Existen 3 formas para establecer la relación

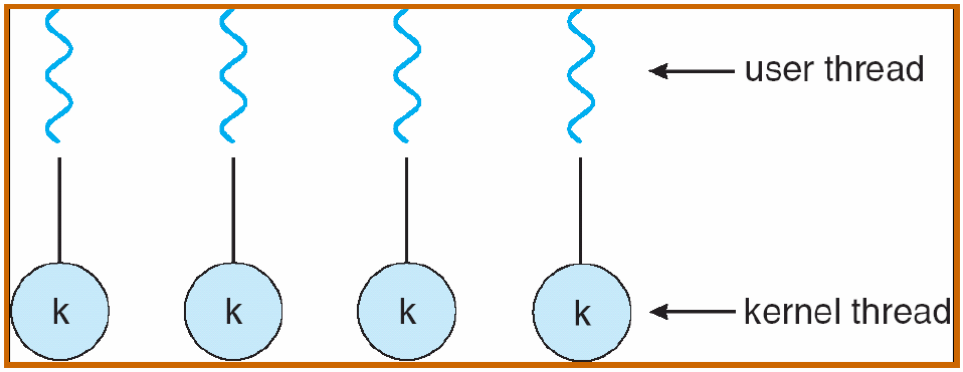
- Modelo Mx1 (Many to one)

El modelo asigna múltiples hilos de usuario a un hilo del kernel.
Este caso se corresponde a los hilos implementados a nivel de usuario, ya que el sistema solo reconoce un hilo de control para el proceso.
Tiene como inconveniente que si un hilo se bloquea, todo el proceso se bloquea.
También, dado que solo un hilo puede acceder al kernel cada vez, no podrán ejecutarse varios hilos en paralelo en múltiples CPUs.



- Modelo 1x1 (one to one)

El modelo asigna cada hilo de usuario a un hilo del kernel. Proporciona una mayor concurrencia que el modelo anterior, permitiendo que se ejecute otro hilo si uno se bloqueó.
Tiene como inconveniente que cada vez que se crea un hilo a nivel de usuario, se crea un hilo a nivel del kernel, y la cantidad de hilos a nivel del kernel están restringidos en la mayoría de los sistemas.



- Modelo MxN (many to many)

El modelo multiplexa muchos hilos de usuario sobre un número menor o igual de hilos del kernel. Cada proceso tiene asignado un conjunto de hilos de kernel, independientemente de la cantidad de hilos de usuario que haya creado.
No posee ninguno de los inconvenientes de los dos modelos anteriores, ya que saca lo mejor de cada uno. El usuario puede crear tantos hilos como necesite y los hilos de kernel pueden ejecutar en paralelo. Asimismo, cuando un hilo se bloquea, el kernel puede planificar otro hilo para su ejecución.
Entonces, el planificador a nivel de usuario asigna los hilos de usuario a los hilos de kernel, y el planificador a nivel de kernel asigna los hilos de kernel a los procesadores.