# HILOS

Básicamente es descomponer una aplicación en varios hilos secuenciales que se ejecutan en cuasi paralelo realizando un proceso más rápido.

Con los hilos agregamos un nuevo elemento: la habilidad de las entidades en paralelo de compartir un espacio y todos sus datos entre ellos (variables globales). Son de 10 a 100 veces más rápidos que los procesos. Lo que agregan los hilos al modelo de procesos es permitir que se lleven a cabo varias ejecuciones en el mismo entorno del proceso. Cuando se ejecuta un proceso con multi-hilamiento en un sistema con una CPU, los hilos toman turnos para ejecutarse como en un proceso y se aprovechan los tiempos muertos. Un hilo puede leer, escribir o incluso borrar la pila de otro hilo.

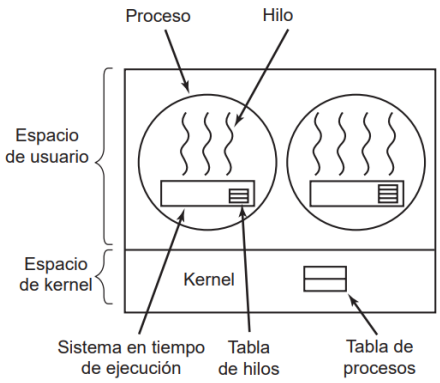
*Lo que estamos tratando de lograr con el concepto de los hilos es la habilidad de que varios hilos de ejecución compartan un conjunto de recursos, de manera que puedan trabajar en conjunto para realizar cierta tarea. Las transiciones entre los estados de los hilos son las mismas que las transiciones entre los estados de los procesos.*

Por lo general, cada hilo llama a distintos procedimientos y, por ende, tiene un historial de ejecución diferente. Esta es la razón por la cual cada hilo necesita su propia pila. Un parámetro para ***pthread_create*** especifica el nombre de un procedimiento para que se ejecute un nuevo hilo, este nuevo hilo se ejecuta en el mismo espacio de direcciones del hilo que lo creo. A menudo todos los hilos son iguales.

- ***pthread_create:*** crea un nuevo hilo.
- ***pthread_exit:*** un hilo termina su trabajo.
- ***pthread_join:*** bloquea al hilo llamador hasta que un hilo especifico haya terminado.
- ***pthread_yield:*** Libera la CPU para dejar que otro hilo se ejecute.

## Implementación de hilos en el espacio de usuario

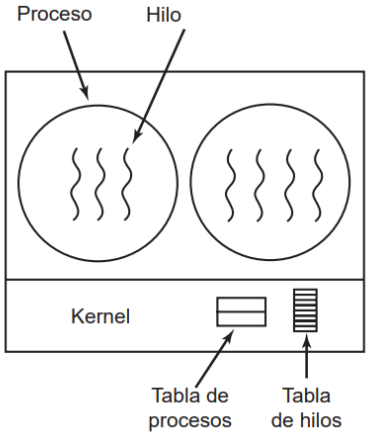
La ventaja de esto es que un paquete de hilos de nivel usuario puede implementarse en un sistema operativo que no acepte hilos. Los hilos se ejecutan encima de un sistema en tiempo de ejecución, el cual es una colección de procedimientos que administran hilos.




- Cada proceso necesita su propia tabla de hilos privada para llevar la cuenta de los hilos en ese proceso.
- La tabla de hilos es administrada por el sistema en tiempo de ejecución.
- La conmutación de hilos es más rápida que una instrucción TRAP.
- El procedimiento que guarda el estado del hilo y el planificador son sólo procedimientos locales.
- Permiten que cada proceso tenga su propio algoritmo de planificación personalizado.

## Implementación de hilos en el kernel

El kernel sabe acerca de los hilos y los administra. No se necesita un sistema en tiempo de ejecución para ninguna de las dos acciones, como se muestra en la figura:




- No tenemos tablas de hilos en cada proceso si no que el kernel tiene una tabla de hilos de todo el sistema.
- Llamada al sistema para crear o destruir hilos en la tabla de hilos del kernel.
- Los hilos del kernel no requieren de nuevas llamadas al sistema no bloqueantes.
- Cuando un hilo se bloquea, el kernel, según lo que decida, puede ejecutar otro hilo del mismo proceso (si hay uno listo) o un hilo de un proceso distinto.

Cuando un hilo se bloquea, el kernel, según lo que decida, puede ejecutar otro hilo del mismo proceso (si hay uno listo) o un hilo de un proceso distinto. Con los hilos de nivel usuario, el sistema en tiempo de ejecución ejecuta hilos de su propio proceso hasta que el kernel le quita la CPU (o cuando ya no hay hilos para ejecutar). Su principal desventaja es que el costo de una llamada al sistema es considerable, por lo que si las operaciones de hilos (de creación o terminación, por ejemplo) son comunes, se incurrirá en una mayor sobrecarga.

## Implementación de Hilos

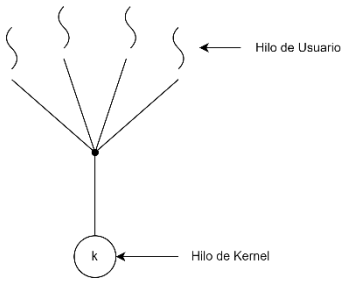
Tipos de implementación; Hilos de Usuario (ULT) y Hilos de Kernel (KLT).

Relación entre ULT y KLT:

- M:1
- 1:1
- M:N (Hibrido)
- 2 niveles

### *Relación Muchos a uno (M:1)*

Muchos a uno significan que en modo usuario yo tengo muchos hilos, pero el kernel solo ve como si fuera un solo hilo o proceso.



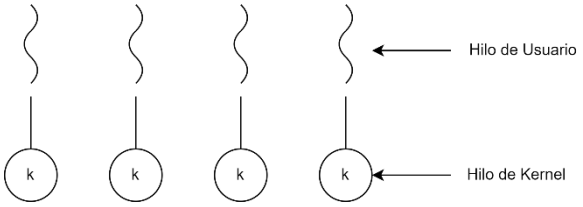

- No necesita soporte de kernel
- Se tiene que liberar voluntariamente un hilo para que se ejecute otro

Si algunos de esos hilos hacen una llamada bloqueante al sistema, para el núcleo todos esos hilos están bloqueados, entonces no puede seguir ejecutándose los otros hilos.

### *Relación uno a uno (1:1) (la mejor aproximación)*

En esta relación el kernel sabe qué hace cada hilo.

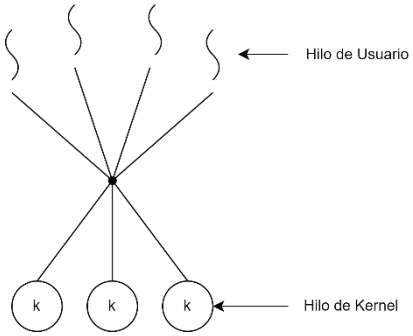
- Mas costosa la creación -> en Kernel (limite cantidad)
- Mayor concurrencia, no se bloquea todo el proceso
- Implementado en la mayoría de los sistemas operativos (hasta XP)




### *Relación Hibrida (M:N)*

No fue muy implementado por su complejidad

- Hilos de Kernel < Hilos de Usuario
- El programador debe especificar cuantos Hilos de Kernel usará.




### *Modelo de 2 niveles*

Implementa M:N y 1:1 simultáneamente, fue usado en algunos sistemas (ej solaris) pero después paso a 1:1.