Hilos

Hilos

- Revisión
- Modelos Multihilados
- Librerías de Hilos
- Aspectos sobre Hilos
- Ejemplos de Sistemas Operativos
- Hilos en Linux

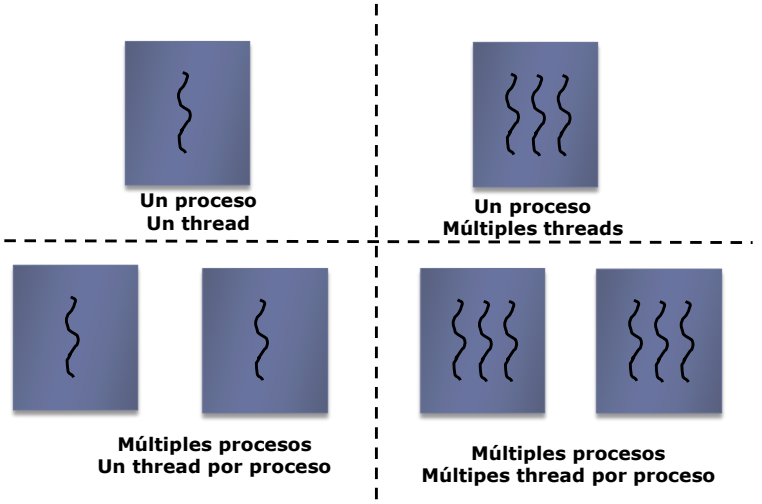
Objetivos

- Introducir la noción de hilo — una unidad fundamental de la utilización CPU que forma la base de los sistemas de computación multihilados
- Discutir las APIs para librerías de hilos Pthreads, Win32 y Java
- Examinar aspectos relacionados a las programación multihilos

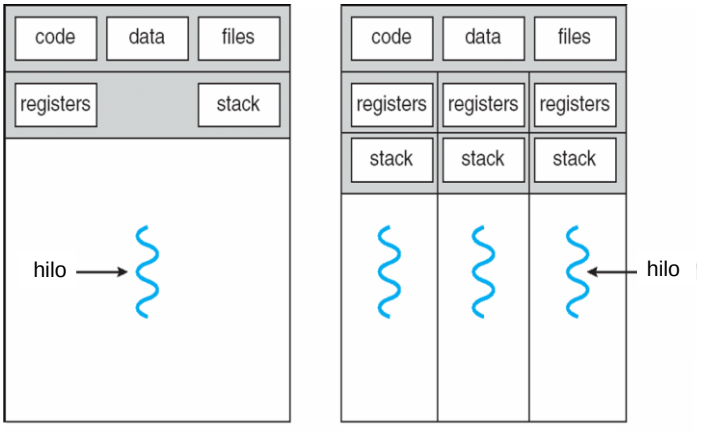
Hilos

- Un *thread* (o *proceso de peso liviano*) es una unidad básica de utilización de CPU, consiste de:
  - contador de programa
  - conjunto de registros
  - espacio de stack
- Un thread comparte con sus threads compañeros su:
  - Sección de código
  - sección de datos
  - recursos del SO.
- Un proceso tradicional o *peso pesado* es igual a una tarea con un solo thread.

Hilos



Hilos



Proceso mono hilado

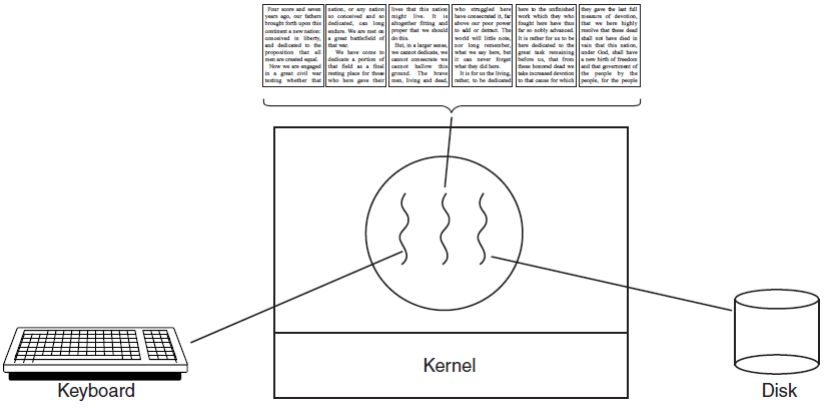
Proceso multihilado

Hilos

- En una tarea con múltiple threads, mientras un thread servidor está bloqueado y esperando, un segundo thread de la misma tarea puede estar corriendo.
- Cooperación de múltiple threads en una misma tarea confiere alto procesamiento total y mejora el rendimiento.
  - Aplicaciones que requieren compartir un buffer común (p.e., productor-consumidor) se benefician con la utilización de threads.
- Los threads proveen un mecanismo que permite a procesos secuenciales hacer llamadas al sistema bloqueantes mientras que también logra paralelismo.

Hilos

- Ejemplo de un procesador de texto con 3 hilos



Beneficios

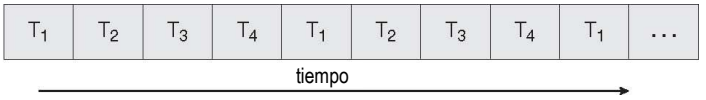
- Capacidad de Respuesta
- Compartir Recursos
  - Dado que los *threads* dentro de un mismo proceso comparten memoria y archivos, pueden comunicarse unos con otros sin invocar al kernel
- Economía
  - Toma menos tiempo crear un nuevo *thread* que un proceso
  - Menos tiempo terminar un *thread* que un proceso
  - Menos tiempo en conmutar entre dos *threads* dentro del mismo proceso
- Utilización de Arquitecturas Multiprocesador
- Escalabilidad

Programación Multicore

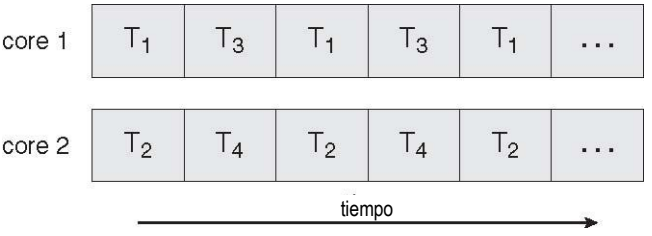
- Los sistemas multicore introducen nuevos desafíos.
  - **Dividir actividades**
  - **Balance**
  - **Partición de datos**
  - **Dependencia de los datos**
  - **Verificación y depuración**

Ejecución Concurrente y Paralela

Un solo núcleo



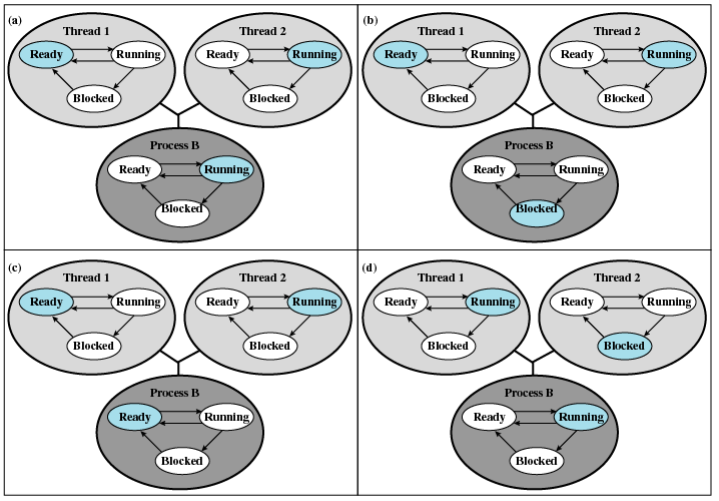
Múltiples núcleos



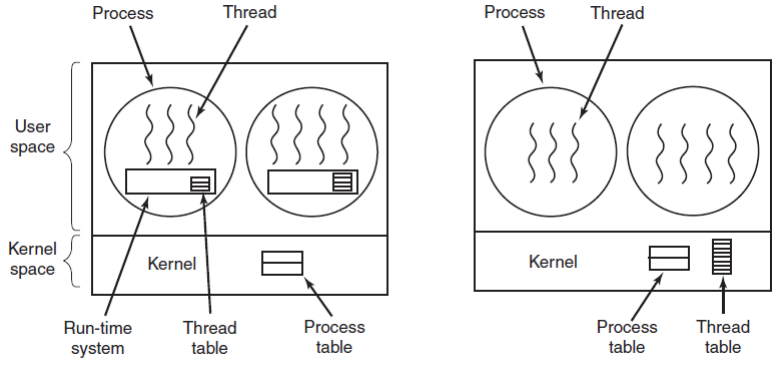
Hilos - Clasificación

- A nivel de **USUARIO** – la administración es realizada por librerías a nivel de usuario.
  - Tres librerías primarias:
    - POSIX Pthreads
    - Win32 threads
    - Java threads
- A nivel de **KERNEL** – la administración es realizada por el sistema operativo
  - Ejemplos:
    - Windows XP/2000/Vista/7/8
    - Solaris (de Sun, ahora Oracle)
    - Tru64 UNIX (de Digital, luego Compaq, finalmente HP)
    - Mac OS X (Apple)

Hilos a Nivel de Usuario – Relación estado Proceso e Hilo



Hilos a Nivel de Usuario y de Kernel

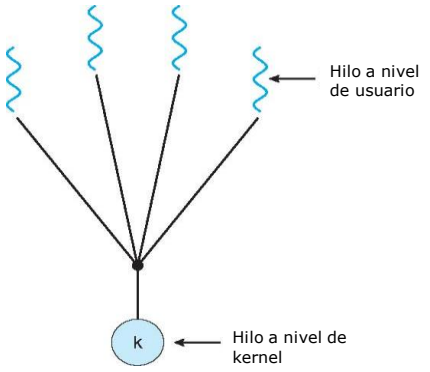


Modelos de Multihilados

- Muchos a Uno (M:1)
- Uno a Uno (1:1)
- Muchos a Muchos (M:M)

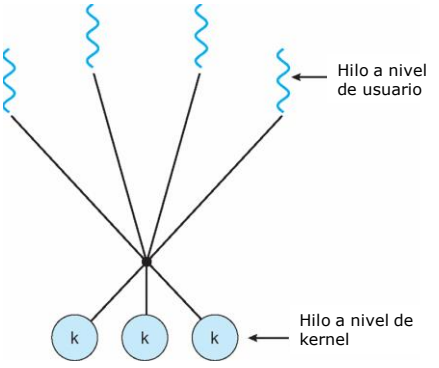
Muchos a Uno

- Muchos hilos a nivel de usuario mapean a un hilo a nivel de kernel.
- Usado en sistemas que no soportan hilos a nivel kernel.



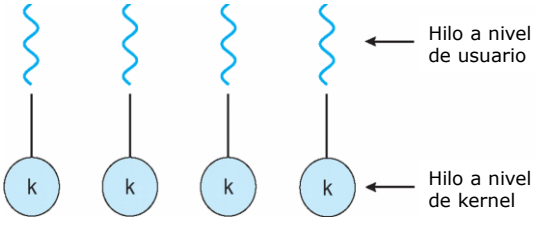
Muchos a Muchos

- Permite que muchos hilos a nivel de usuario mapeen a muchos hilos a nivel de kernel
- Permite al SO crear un número suficiente de hilos a nivel de kernel
- Solaris antes de la versión 9
- Windows NT/2000 en adelante con paquete *ThreadFiber*



Uno a Uno

- Cada thread nivel usuario mapea a un thread kernel.
- Ejemplos
  - Windows NT/XP/2000 y los que siguen
  - Linux
  - Solaris 9 y los que siguen



Librerías de Hilos

- Las librerías de hilos proveen a los programadores con APIs para crear y administrar hilos
- Dos formas primarias de implementarlas
  - Librerías enteramente en espacio de usuario
  - Librería a nivel de Kernel soportada por el SO

Librería de hilos - Pthreads

- Pueden ser provistas sea a nivel de usuario como a nivel de kernel
- Es un standard POSIX (IEEE 1003.1c) API para creación y sincronización de hilos
- Las API especifican el comportamiento de la librería de hilos
- Común en SOs UNIX (Solaris, Linux, Mac OS X)

Cancelación de Hilos

- Terminar un hilo antes que finalice
- Dos propuestas generales:
  - **Cancelación asincrónica** termina el hilo señalado inmediatamente
  - **Cancelación Diferida** permite al hilo señalado verificar periódicamente si debería ser cancelado

Manejo de Signal

- Los Signals son usados en UNIX para notificar a un proceso que un particular evento ha ocurrido
- Un signal handler es usado para signals a procesos
  1. El Signal es generado por un particular evento
  2. El Signal es enviado a un proceso
  3. El Signal es manejado
- Opciones:
  - Enviar el signal al hilo sobre el cual el signal se aplica
  - Enviar el signal a cada hilo en el proceso
  - Enviar el signal a ciertos hilos en el proceso
  - Asignar un hilo específico para recibir todos los signals al proceso

Pools de Hilos

- Crea un número de hilos en un pool donde esperan por trabajo
- Ventajas:
  - Usualmente es ligeramente mas rápido servir un requerimiento con un hilo existente que crear uno nuevo
  - Permite que el número de hilos de la aplicación sea liitado al tamaño del pool

Hilos en Linux

- Linux se refiere a ellos como *tareas* más que como *hilos*
- La creación de hilos es hecha por la llamada a sistema **clone()**
- **clone()** permite a una tarea hija compartir el espacio de direcciones de la tarea-padre (proceso)

Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "*Operating System Concepts*", 7ma Edición 2009; 9na Edición 2012; 10ma Edición 2018.
- Stallings, W. *"Operating Systems: Internals and Design Principles"*, Prentice Hall, 5ta Edición 2005; 6ta Edición 2009; 7ma Edición 2011, 9na Edición 2018.
- Tanenbaum, A.; "*Modern Operating Systems*", Addison-Wesley, 3ra Edición 2008, 4ta. Edición 2014.