Sistemas Operativos

UNIDAD 3 – SISTEMA DE PROCESOS

THREADS – HILOS (PROCESOS LIGEROS)

Índice

- Dos visiones de un proceso → Hilos
- Sistema Multihilo
- Estados de los Hilos
- Implementación de los Hilos
  - User Level Threads, ULT
  - Kernel Level Threads, KLT

Dos visiones de un proceso

Un proceso :

- Incluye un **espacio de direcciones virtuales** para mantener la imagen del proceso
- Mantiene una ejecución secuencial de sus instrucciones que puede ser intercalada, en el procesador, con la de otros procesos.

El proceso es visto por el Sistema Operativos como:

- La unidad propietaria de los recursos
- La unidad expedición (aquello que se asigna al procesador)

Estas dos características son tratadas por el S.O. de manera independiente.

- **La unidad propietaria se conoce como Proceso** (Process)
- **La unidad de asignación se conoce como Hilo** (Thread)

Información global para cada proceso	Información local para cada hilo
Espacio direcciones global (imagen del proceso)	Estado de ejecución (ejecución, preparado, espera, ..)
Variables globales	Variables locales
Código (instrucciones)	Registros de datos (AX, BX, CX, ...)
Recursos utilizados (ficheros abiertos, etc.)	Registros de control y estado (CP, Z, C, ...)
Procesos hijos	Pilas de llamadas

Cada hilo tiene:

- Un estado de ejecución (Listo, Bloqueado, etc.) independiente del estado de ejecución del proceso.
- El contexto del procesador que el S.O. guarda cuando el hilo no ejecuta. Asociado al hilo en ejecución.
- Tiene una pila de ejecución propia.
- Tiene almacenamiento estático para las variables locales, independiente de la del proceso.
- Tiene acceso a memoria, variables y recursos asignados al proceso.
- Las variables y recursos globales del proceso (**compartidas por todos los hijos**) deben ser protegidas en los hilos con mecanismos de sincronización y exclusión mutua.

Número de hilos

En un proceso puede haber 1 o más hilos.

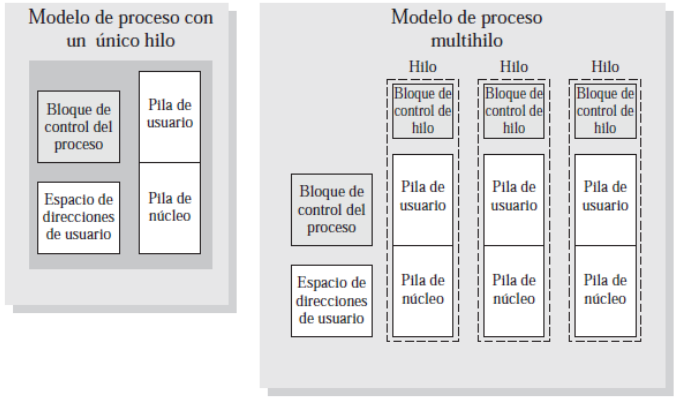


Figura 4.2. Modelos de proceso con un único hilo y multihilo.

Dos visiones de un proceso

Un proceso :

- Incluye un **espacio de direcciones virtuales** para mantener la imagen del proceso
- Mantiene una ejecución secuencial de sus instrucciones que puede ser intercalada, en el procesador, con la de otros procesos.

El proceso es visto por el Sistema Operativos como:

- La unidad propietaria de los recursos
- La unidad expedición (aquello que se asigna al procesador)

Estas dos características son tratadas por el S.O. de manera independiente.

- **La unidad propietaria se conoce como Proceso** (Process)
- **La unidad de asignación se conoce como Hilo** (Thread)

Información global para cada proceso	Información local para cada hilo
Espacio direcciones global (imagen del proceso)	Estado de ejecución (ejecución, preparado, espera, ..)
Variables globales	Variables locales
Código (instrucciones)	Registros de datos (AX, BX, CX, ...)
Recursos utilizados (ficheros abiertos, etc.)	Registros de control y estado (CP, Z, C, ...)
Procesos hijos	Pilas de llamadas

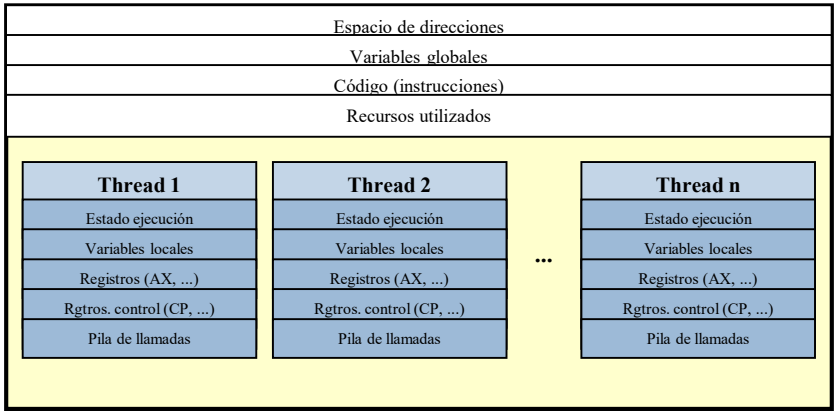
Cada hilo tiene:

- Un estado de ejecución (Listo, Bloqueado, etc.) independiente del estado de ejecución del proceso.
- El contexto del procesador que el S.O. guarda cuando el hilo no ejecuta. Asociado al hilo en ejecución.
- Tiene una pila de ejecución propia.
- Tiene almacenamiento estático para las variables locales, independiente de la del proceso.
- Tiene acceso a memoria, variables y recursos asignados al proceso.
- Las variables y recursos globales del proceso (**compartidas por todos los hijos**) deben ser protegidas en los hilos con mecanismos de sincronización y exclusión mutua.

Número de hilos

En un proceso puede haber 1 o más hilos.

Todos los hilos de un proceso comparten el estado y recursos del proceso, residen en el mismo espacio de direcciones y tienen acceso a los mismos datos.



Un sistema multihilo

Multihilo: Capacidad del S.O. de soportar múltiples hilos de ejecución en un solo proceso

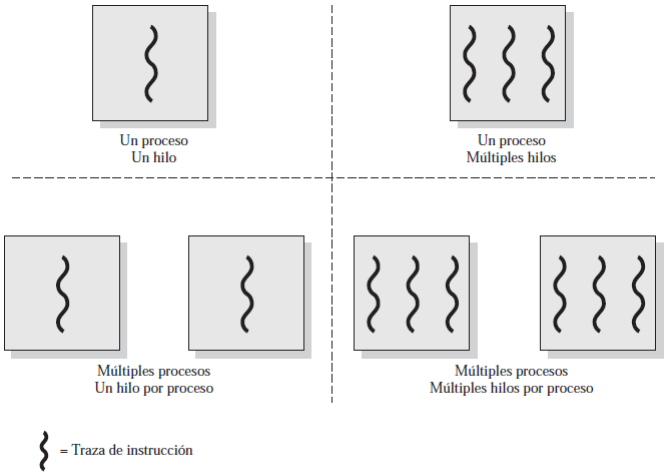


Figura 4.1. Hilos y procesos [ANDE97].

Los hilos de un proceso comparten el estado y los recursos del proceso.
Si un hilo:

- modifica una variable → los demás hilos ven la modificación
- abre para lectura un fichero → los demás hilos podrán también leer

Rendimiento

1. Lleva menos tiempo crear un hilo (en un proceso existente) que crear un proceso nuevo. *(aprox 10x)*
2. Lleva menos tiempo finalizar un hilo
3. Lleva menos tiempo cambiar entre dos hilos del mismo proceso
4. Se pueden comunicar entre hilos sin invocar al núcleo (accediendo a su memoria/variables común)

Ejemplos S.O.

- **MS-DOS:**
  Un solo proceso, un solo hilo/proceso.
- **UNIX** (algunas variedades):
  Múltiples procesos, un solo hilo/proceso
- **Windows, Solaris, Mach, OS/2, Linux**:
  Múltiples procesos, múltiples hilos/proceso

¿ Son útiles en sistemas mono-usuario/multiproceso?

- Foreground/background tasks:
  Edición/Rev.Ortográfica, Edición/CalculoHoja, etc…
- Procesamiento asíncrono:
  Salvado automático asíncrono, el hilo de backup tiene acceso a todo, datos, ficheros, etc… No hace falta sincronizarlos.
- Velocidad de ejecución:
  Computo de datos mientras lee los siguientes, el proceso no se bloquea por la lectura.
- Estructura modular:
  Un diseño más fácil por tareas, acciones, fuentes de datos, …

La planificación y la activación se realiza a nivel de hilo
La suspensión y finalización a nivel de proceso

Estados de los hilos

Los principales estados de un hilo son:
Ejecución, Listo y Bloqueado

Cambio de estado del Hilo: Operaciones básicas

Creación:

Al crear un proceso se crea un hilo principal.

Se crea un nuevo hilo desde el hilo principal: *(los hilos "hijo" también pueden crear hilos)*

Se pasa : Puntero a su código, Argumentos

El nuevo hilo tendrá un espacio de contexto y un espacio de pila.

El nuevo hilo pasa a Listo

Bloqueo

A la espera de un suceso se guardan registros, PC y punteros de pila.

El hilo pasa a la cola de bloqueados del suceso/dispositivo relacionado.

El procesador puede seleccionar otro hilo (Listo) del mismo u otro proceso.

Desbloqueo

Al ocurrir el suceso, el hilo bloqueado pasa a la cola de Listos

Terminación:

Se liberan el contexto y las pilas.

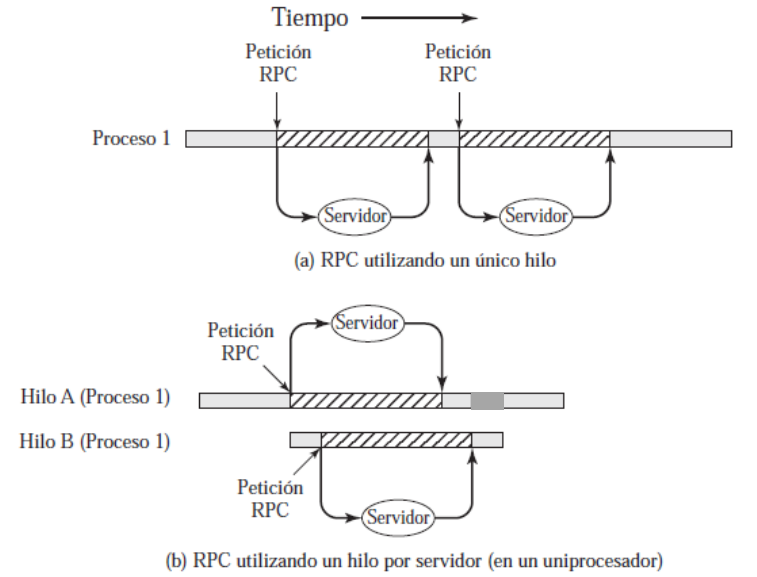


Figura 4.3. Llamadas a Procedimiento Remoto (RPC) utilizando hilos.

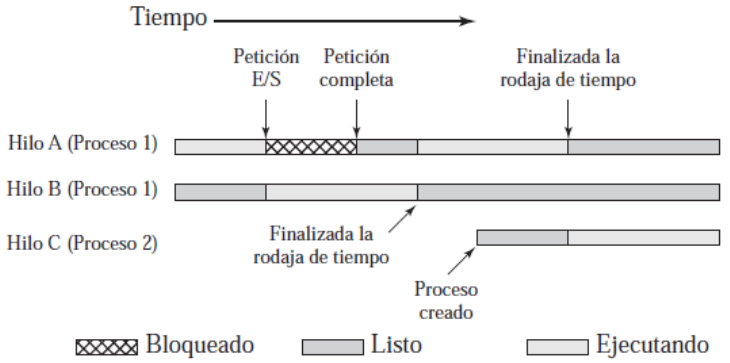


Figura 4.4. Ejemplo multihilo en un uniprocesador.

Implementación de los hilos

Los hilos se pueden implementar

- A nivel de usuario (User Level Threads, ULT)
- A nivel de kernel (Kernel Level Threads, KLT) (Kernel-Supported-Threads)(Lightweight processes)

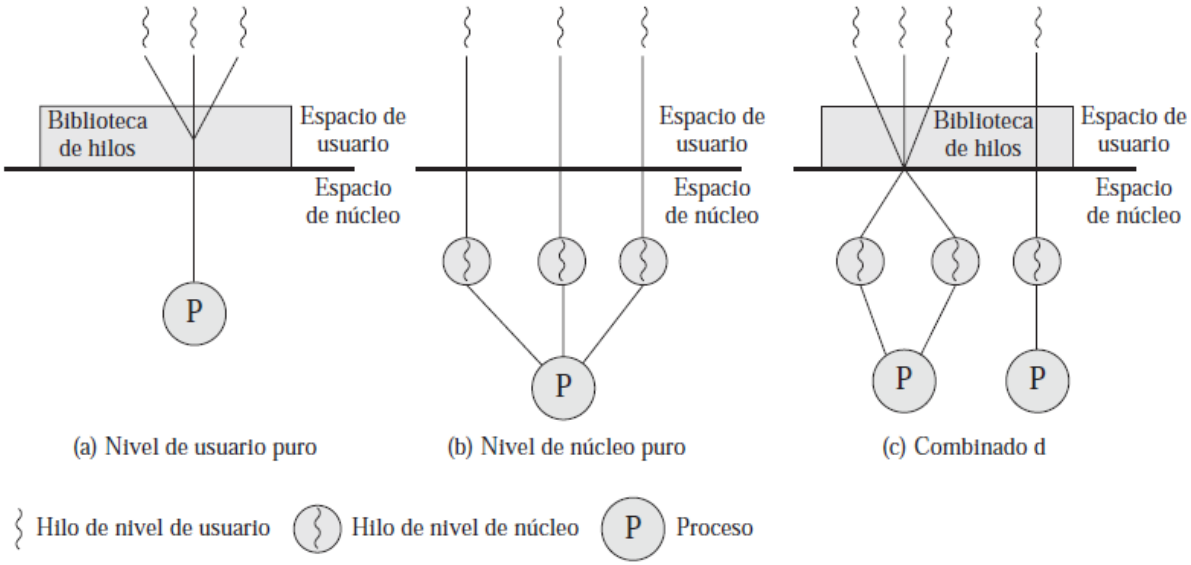
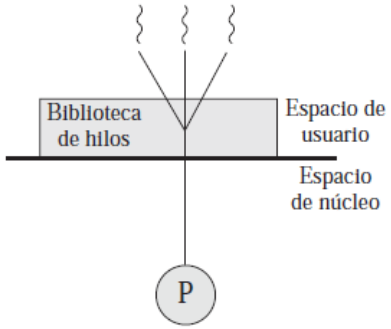


Figura 4.6. Hilos de nivel de usuario y de nivel de núcleo.

Implementación de los Hilos – User Level Threads



(a) Nivel de usuario puro

- Toda la gestión de hilos en la aplicación → El núcleo no es consciente de su existencia.
- El programador hará uso de la biblioteca de hilos, que ofrece funciones para:
  crear, destruir, sincronizar, planificar, comunicar hilos (mensajes), salvar y restaurar hilos.
  - La aplicación arranca con un único hilo,
  - Este puede crear otro hilo mediante una función de librería,
  - La librería toma el control (llamada a procedimiento), reserva espacio y crea las estructuras de datos necesarias,
  - La librería planifica la ejecución de un nuevo hilo en función de una política concreta establecida
  - La librería retorna el control de ejecución al hilo elegido de entre los que estaban en Listo.
  - Cuando la librería toma el control, salva el contexto del hilo en ejecución para restaurarlo cuando cede el control al mismo.
  - Todas estas operaciones se realizan en el espacio de usuario dentro del mismo proceso.
  - El núcleo desconoce la existencia de los hilos, planifica el proceso como un todo.

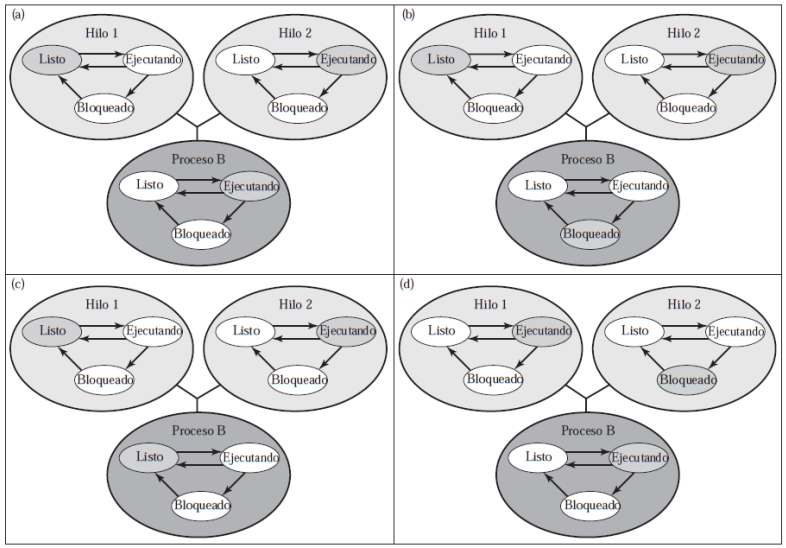
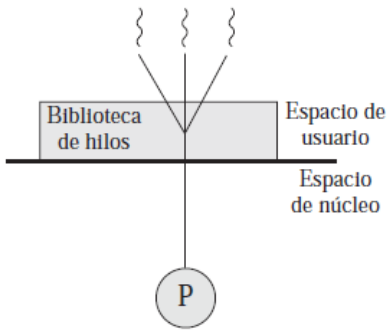


Figura 4.7 Ejemplos de relaciones entre los estados de los hilos de nivel de usuario y los estados de proceso.

Un proceso B tiene dos hilos, estando ejecutando el Hilo2.
Analicemos varias situaciones.

1. Hilo 2 ejecuta E/S. Fig.(a) -> Fig.(b). Hilo2 no está ejecutando realmente
2. Fin de quantum proceso B -> Fig.(a) -> Fig.(c).
3. Hilo 2 necesita acción del Hilo1 (o del proceso B). Fig.(a)-> Fig.(d)

Implementación de los Hilos – User Level Threads



(a) Nivel de usuario puro

- Toda la gestión de hilos en la aplicación → El núcleo no es consciente de su existencia.
- El programador hará uso de la biblioteca de hilos, que ofrece funciones para:
  crear, destruir, sincronizar, planificar, comunicar hilos (mensajes), salvar y restaurar hilos.
  - La aplicación arranca con un único hilo,
  - Este puede crear otro hilo mediante una función de librería,
  - La librería toma el control (llamada a procedimiento), reserva espacio y crea las estructuras de datos necesarias,
  - La librería planifica la ejecución de un nuevo hilo en función de una política concreta establecida
  - La librería retorna el control de ejecución al hilo elegido de entre los que estaban en Listo.
  - Cuando la librería toma el control, salva el contexto del hilo en ejecución para restaurarlo cuando cede el control al mismo.
  - Todas estas operaciones se realizan en el espacio de usuario dentro del mismo proceso.
  - El núcleo desconoce la existencia de los hilos, planifica el proceso como un todo.

Ventajas de usar ULT en vez de KLT

- El cambio de hilo no necesita privilegios de kernel, los metadatos necesarios están en espacio de usuario. El proceso no debe cambiar a modo kernel. Se evita la sobrecarga de cambio de modo
- Se puede realizar planificaciones específicas y a medida, en función de la aplicación (round-robin, prioridades, etc…)
- Los ULT pueden ejecutar en cualquier S.O. *(no necesario núcleo especial)*
  La biblioteca de hilos es compartida por todas las aplicaciones.

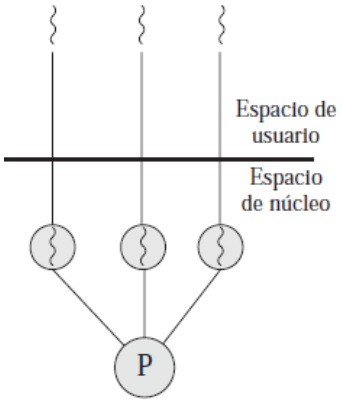
Desventajas de usar ULT en vez de KLT

- En un S.O. la mayoría de las llamadas al sistema son bloqueantes.
  Si un hilo hace una llamada bloqueante, el S.O. bloquea todo el proceso (todos los hilos)
- Una estrategia ULT no puede aprovechar las ventajas de un sistema multiprocesador. El kernel asigna el proceso completo a un único procesador.

¿Cómo superar las desventajas?

- Diseñando la aplicación con múltiples procesos → Perdemos la ventaja de cambio de modo.
- Mediante recubrimiento (**jacketing**). Se crean rutinas que realizan la llamada al sistema sólo si el dispositivo E/S está disponible, si no lo está, no realiza la llamada y se pone el hilo en Listo y cede el control (del hilo).
  Cuando le toque al hilo de nuevo, vuelve a comprobar de nuevo el dispositivo.

Implementación de los Hilos – Kernel Level Threads



(b) Nivel de núcleo puro

- Todo el trabajo de gestión de los hilos lo realiza el núcleo.
- En el código de la aplicación no hay código de gestión de hilos, sólo llamadas al API del núcleo para la gestión de hilos (W2K, Linux y OS2)
- Se puede programar cualquier aplicación como multihilo, donde todos los hilos pertenecen al mismo proceso.
- Cualquier aplicación puede programarse multihilo.
- El kernel mantiene la información del proceso (como un todo) y de todos los hilos del proceso
- El kernel realiza la **planificación a nivel de hilo**, incluso en múltiples procesadores y sin penalizar el resto de hilos si uno de ellos se bloquea

Ventajas de usar KLT en vez de ULT

- Se pueden usar múltiples procesadores
- No se bloquean todos los hilos de un proceso si uno se bloquea
- Las rutinas del kernel suelen ser multihilo

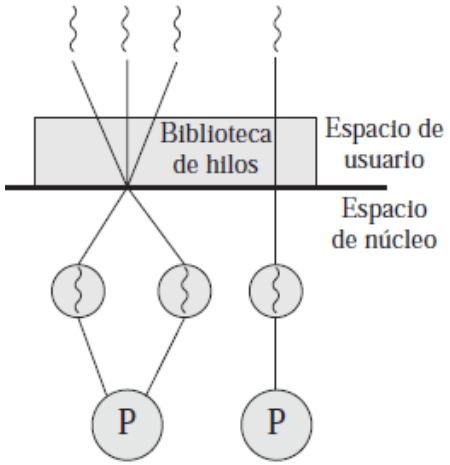
Desventajas de usar KLT en vez de ULT

- Para cambiar de un hilo a otro dentro del mismo proceso necesitamos un cambio a modo kernel.

Tabla 4.1. Latencia de las Operaciones en Hilos y Procesos (μS) [ANDE92].

Operación	Hilos a nivel de usuario	Hilos a nivel de núcleo	Procesos
Crear Proceso Nulo	34	948	11.300
Señalizar-Esperar	37	441	1.840

Implementación de los Hilos – KLT & ULT Combinados



(c) Combinado d

- Algunos S.O. ofrecen una aproximación combinada (Solaris)
- Permiten asociar a uno o varios ULT un KLT
- Una llamada bloqueante no bloquea todos los ULT, sólo los asociados al mismo KLT
- Se generan varios modelos de asociación, 1:1, M:1 y M:M
- El programador puede ajustar el número de KLT para cada aplicación y vincular qué ULT va con qué KLT.
- Una aplicación correctamente diseñada puede combinar las ventajas de ambos modelos.

Relaciones entre Hilos y Procesos

Varios tipos de relación entre la unidad de Activación (hilos) y la Unidades de Asignación (procesos)

Lo normal es pensar en múltiples hilos en un único proceso (Relación Muchos-a-uno)

Tabla 4.2. Relación Entre Hilos y Procesos.

Hilos:Procesos	Descripción	Sistemas de Ejemplo
1:1	Cada hilo de ejecución es un único proceso con su propio espacio de direcciones y recursos.	Implementaciones UNIX tradicionales.
M:1	Un proceso define un espacio de direcciones y pertenencia dinámica de recursos. Se pueden crear y ejecutar múltiples hilos en ese proceso.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH.
1:M	Un hilo puede migrar de un entorno de proceso a otro. Esto permite a los hilos moverse fácilmente entre distintos sistemas.	Ra (Clouds), Emerald.
M:N	Combina atributos de M:1 y casos 1:M.	TRIX.

Fin

UNIDAD 3 – SISTEMA DE PROCEOSS

THREADS – HILOS (PROCESOS LIGEROS)