Hilos (Threads)

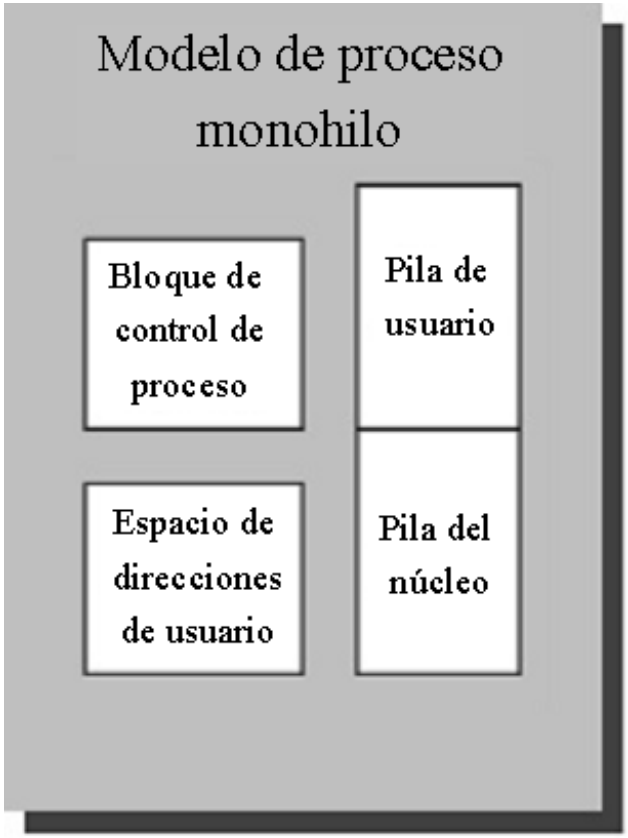
Dos características de un proceso

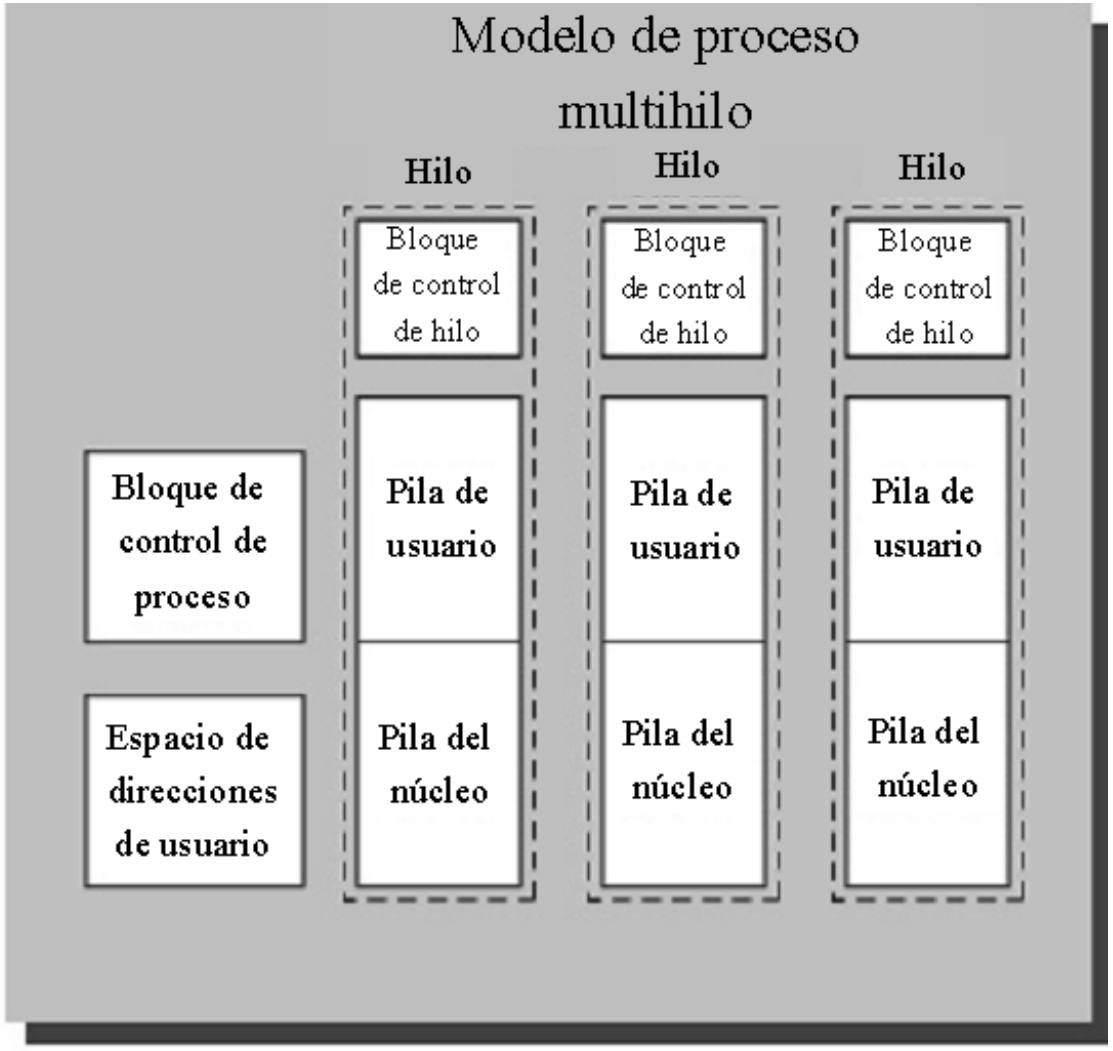
- Como Unidad de propiedad de los recursos:
  - El proceso incluye un espacio de direcciones virtuales para mantener la imagen del proceso.
- Como Unidad de expedición:
  - Sigue un camino de ejecución que puede ser intercalada con la de otros procesos.
- Estas dos características son tratadas de manera independiente por el sistema operativo.
  - **La unidad de expedición se conoce como hilo.**
  - **La unidad de propiedad de los recursos se conoce como proceso o tarea.**

Proceso vs. Hilo en entorno multihilo

- Proceso es la unidad de protección y asignación de recursos
  - Tiene un espacio de direcciones virtuales, que contiene la imagen del proceso.
  - Acceso protegido a los procesadores, a otros procesos (comunicación entre procesos), archivos y a recursos de E/S (dispositivos y canales).
- En un proceso puede haber uno o más Hilos. Cada hilo tiene:
  - Un estado de ejecución (Ejecución, Listo, etc.) independiente
  - El contexto del procesador que se salva cuando no está ejecutando se asocia al Hilo en ejecución.
  - Tiene una pila de ejecución.
  - Almacenamiento estático para las variables locales.
  - Acceso a la memoria, variables y recursos del proceso, compartidos con todos los hilos del mismo.
  - Las variables globales de un proceso deben ser protegidas en los hilos

Monohilo - Multihilo



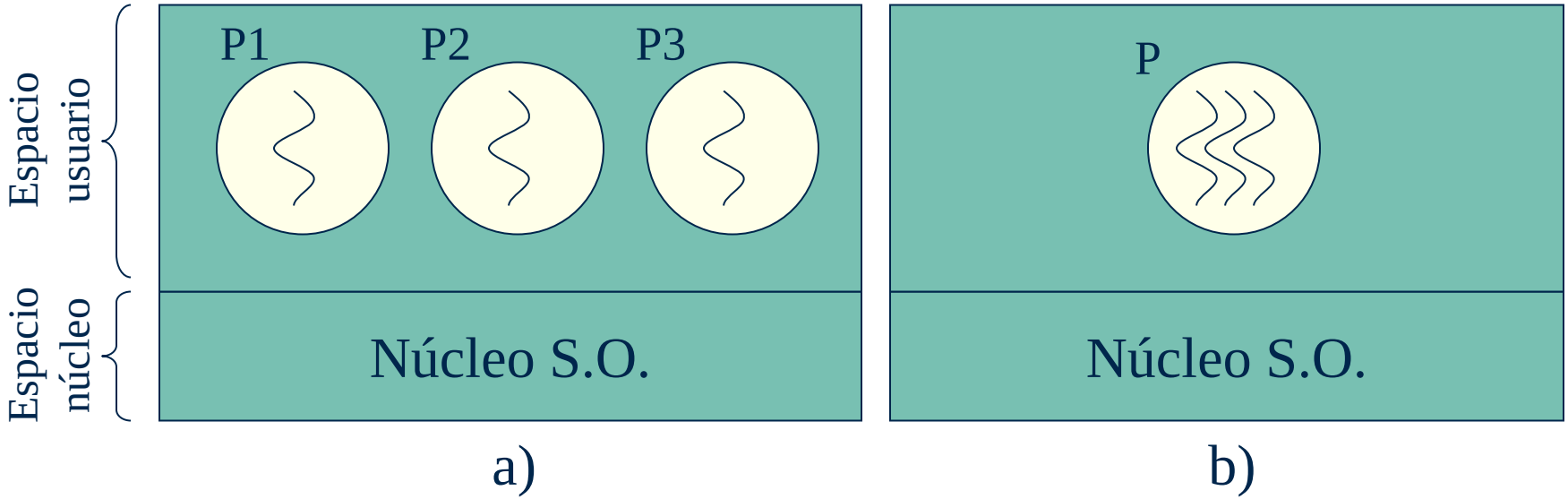


Multihilo

- Sistema operativo que mantiene varios hilos de ejecución dentro de un mismo proceso.
- MS-DOS soporta un solo hilo.
- UNIX soporta múltiples procesos de usuarios, pero sólo un hilo por proceso.
- Windows 2000, Solaris, Linux, Mach, y OS/2 soportan múltiples hilos.

Hilos

- Hilos (threads) o “procesos ligeros”
  - Son unidades de trabajo (expedición) dentro de un proceso



a) 3 procesos con 1 hilo (3 pesos pesados)

b) 1 sólo proceso con 3 hilos (3 pesos ligeros)

Gestión de Procesos - Hilos

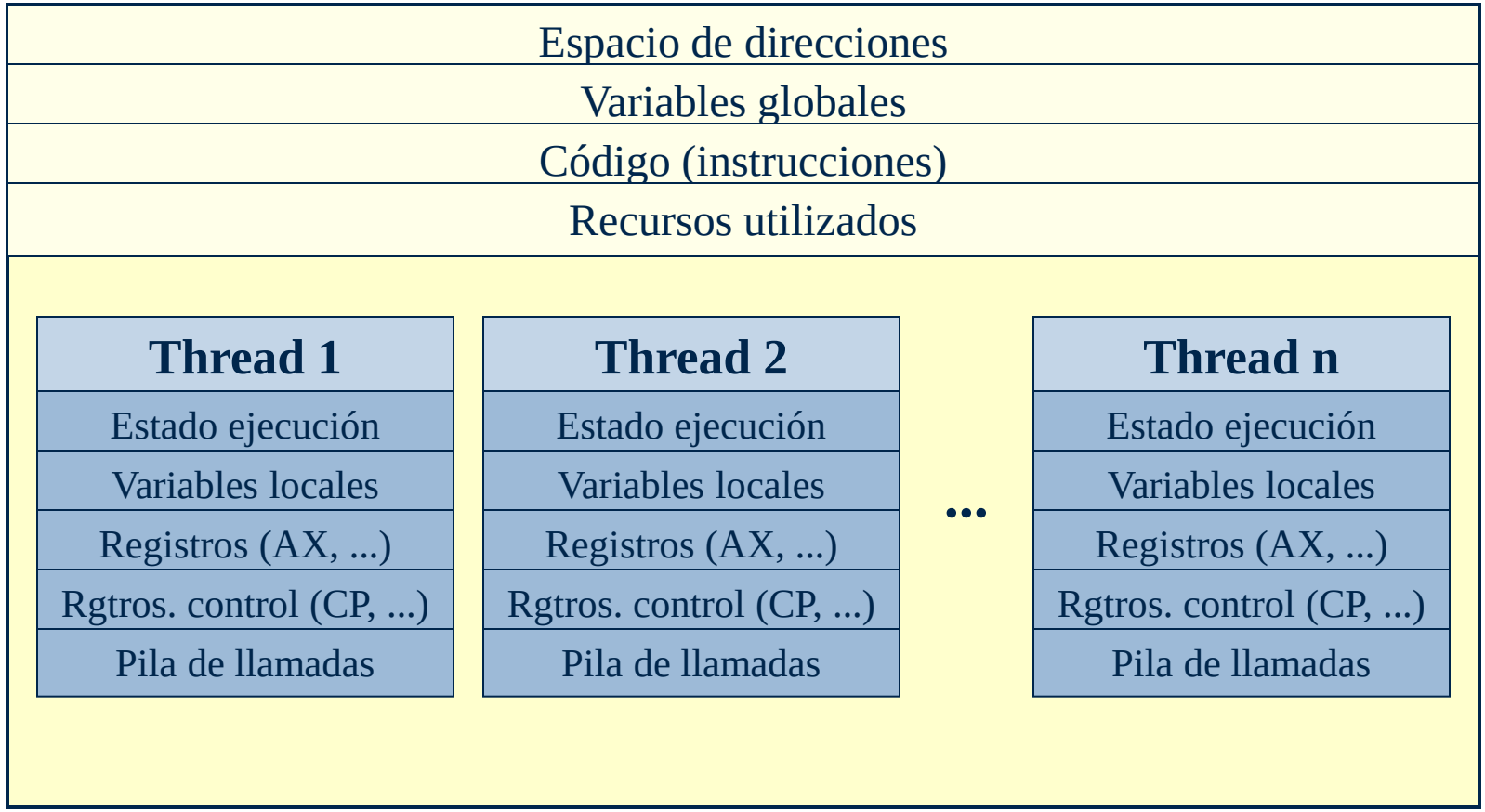
- Su gestión es similar a la de los procesos: cada hilo tendrá un estado de ejecución (ejecución, preparado, etc.).
- Cada hilo puede estar en un estado diferente, pero cuando un proceso cambia de estado (suspensión o terminación), todos los hilos deberán cambiar.
- Un proceso contiene información global compartida por todos los hilos, pero además, información específica para cada tarea.

Información necesaria:

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

Gestión de Procesos - Hilos

- Esquema de un proceso con hilos:



Multihilo

Ejemplos de uso de los hilos en un sistema monousuario y multiproceso:

- Trabajo interactivo y en segundo plano (ej: comprobar ortografía y atender usuario)
- Procesamiento asíncrono y repetitivo (ej: planificar backups con el S.O.)
- Aceleración de la ejecución (ej: procesamiento y E/S solapadas)
- Estructuración modular de los programas

Beneficios derivados del uso de hilos de ejecución:

- Mayor control para el programador
- Programas más fáciles de diseñar e implementar
- Menos tiempo de creación de un nuevo hilo en un proceso que un nuevo proceso
- Menos tiempo en terminar un hilo que un proceso
- Menos tiempo en cambiar entre hijos de un mismo proceso que cambiar de proceso
- Aumentan la eficiencia de comunicación entre programas en ejecución.

  Para comunicar procesos entre sí es necesaria la intervención del nucleo via primitivas de comunicación.

  Para comunicar hilos entre sí no hace falta la intervención del nucleo, ya que comparten memoria y archivos.

- En sistemas multiprocesador, varios hilos pueden ejecutar en concurrencia real en distintos procesadores.
- En un sistema monoprocesador, la concurrencia de los hilos es lógica.

Inconvenientes derivados del uso de hilos de ejecución:

- Necesidad de exclusión mútua en el:
  - Acceso a recursos (ficheros, canales E/S, etc…)
  - Memoria compartida del proceso vista por todos los hilos

Estados de un Hilo

- Los principales estados de un hilo son:
  - Ejecucíon, Listo y Bloqueado
- Hay cuatro operaciones básicas relacionadas con el cambio de estado en hilos:
  - Creación:
    - Al crear un proceso se crea un hilo principal.
    - Se crea un nuevo hilo desde el hilo principal. El nuevo hilo pasa a Listo
  - Bloqueo
    - A la espera de un suceso se guardan registros, PC y punteros de pila. El proceso pasa a la cola de bloqueados del suceso/dispositivo relacionado.
  - Desbloqueo
    - Al ocurrir el suceso el hilo bloqueado pasa a la cola de Listos
  - Terminación:
    - Se liberan el contexto y las pilas.

Ejemplo: RPC contra distintos servidores

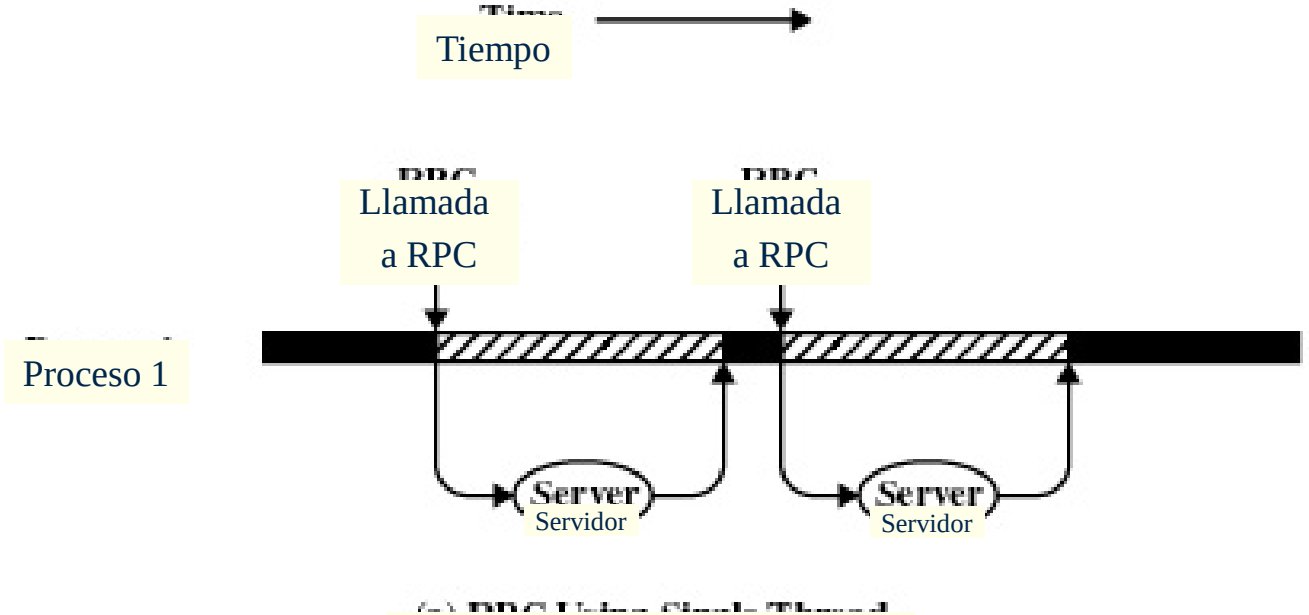
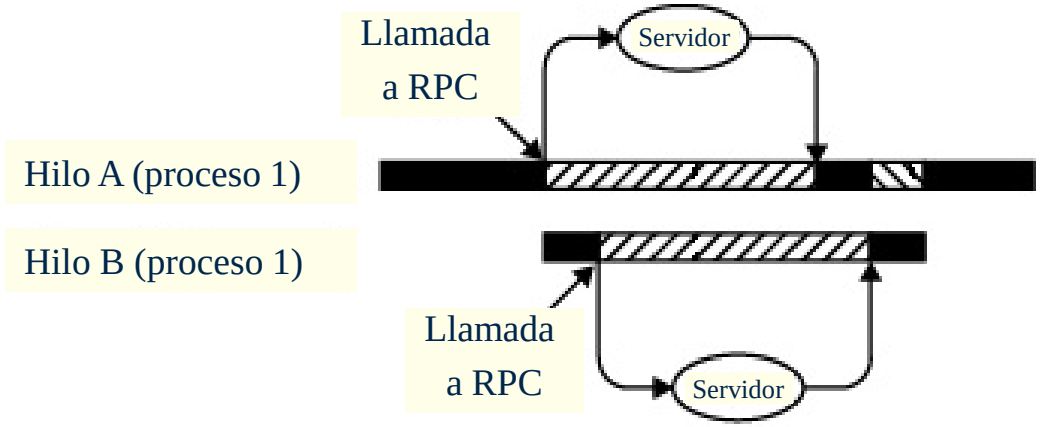


Figure 4.3 Remote Procedure Call (RPC) Using Threads

Ejemplo: RPC contra distintos servidores



(b) RPC utilizando un hilo por servidor (en un monoprocesador)

Bloqueado, esperando respuesta de una RPC

Bloqueado, esperando al procesador que está siendo usado por el hilo B

Ejecutando

Figure 4.3 Remote Procedure Call (RPC) Using Threads

Ejemplo: Multithreading en monoprocesador

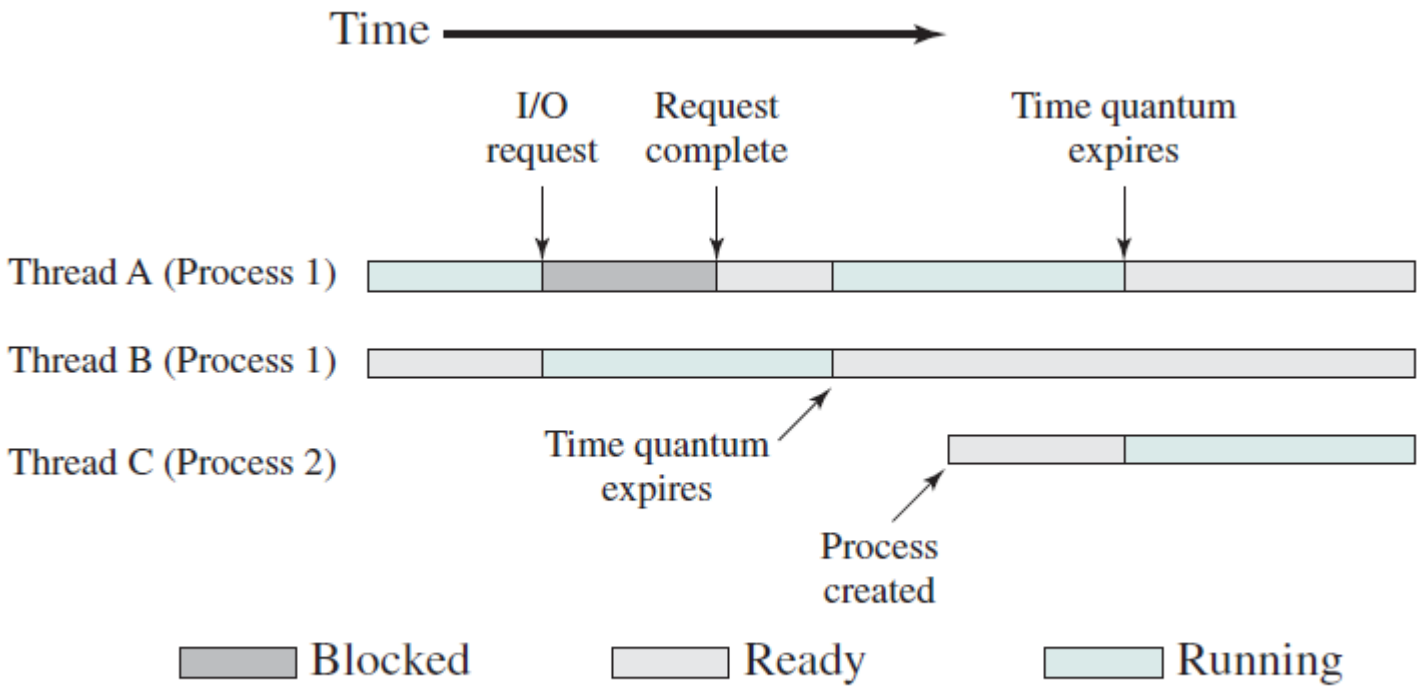


Figure 4.4 Multithreading Example on a Uniprocessor

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)

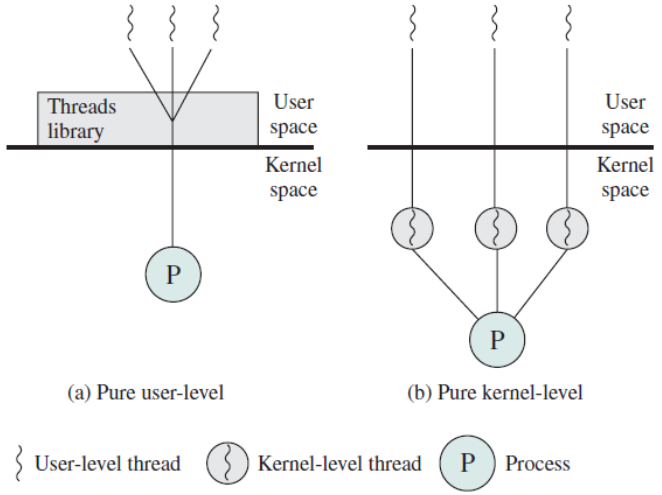


Figure 4.5 **User-Level and Kernel-Level Threads**

ULT

- Todo el trabajo de gestión de los hilos lo realiza la aplicación y el nucleo no es consciente de la existencia de hilos.
- El programador hará uso de la biblioteca de hilos que ofrece funciones para crear, destruir, sincronizar, planificar, comunicar hilos, 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, reserva espacio y crea las estructuras 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.

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)

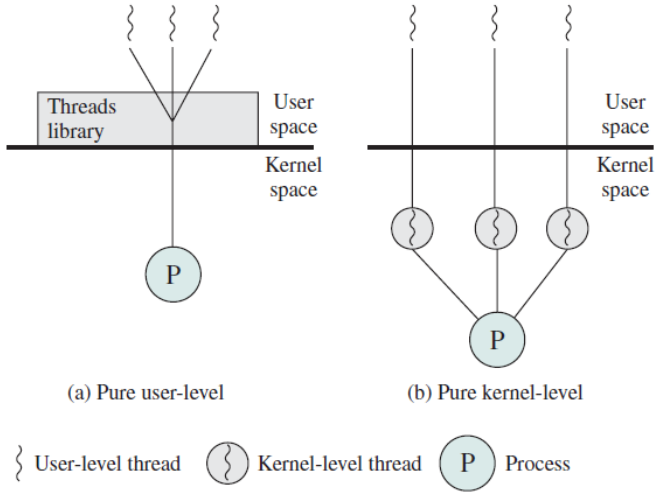


Figure 4.5 **User-Level and Kernel-Level Threads**

KLT

- 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.
- El kernel mantiene la información del proceso (como un todo) y de todos los hilos del proceso
- El kernel realiza la planificación, incluso en múltiples procesadores y sin penalizar el resto de hilos si uno de ellos se bloquea

Implementación de los hilos

Ventajas de usar ULT en vez de KLT

- El intercambio entre hilos no necesita privilegios de kernel, por lo que el proceso no debe cambiar a modo kernel para gestionar los hilos. Se evita la sobrecarga de cambio de modo
- Se puede realizar planificaciones específicas en función de la aplicación (round-robin, prioridades, etc…)
- Los ULT pueden ejecutar en cualquier S.O. 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, por lo que 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 a un único procesador.

¿Cómo superar estos inconvenientes?

- 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. Cuando le toque al hilo de nuevo, vuelve a comprobar antes el dispositivo.

Implementación de los hilos

Ventajas de usar KLT en vez de ULT

- Se pueden usar multiples procesadores
- No se bloquean todos los hilos de un proceso si uno se bloquea

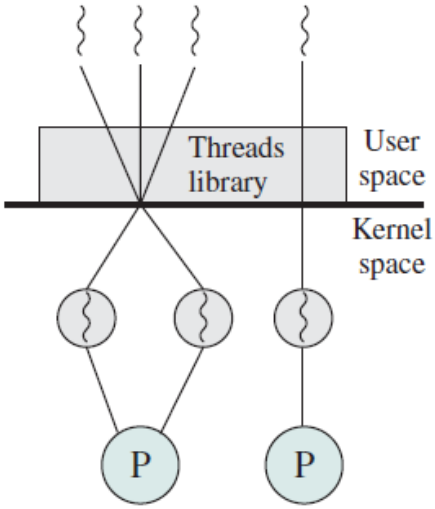
Desventajas de usar KLT en vez de ULT

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

Implementación de los hilos

Aproximaciones combinadas

- Algunos S.O. ofrecen una aproximación combinada (Solaris)
- Permiten asociar a uno o varios ULT un 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.



(c) Combined

Kernel vs. User-Level Threads

Multithreading Models