Hilos

Pedro O. Pérez M., PhD.

Diseño de sistemas embebidos avanzados
Tecnológico de Monterrey

pperezm@tec.mx

08-2023

- El modelo de proceso introducido anteriormente asumía que un proceso era un programa en ejecución con un solo hilo de control. Sin embargo, prácticamente todos los sistemas operativos modernos proporcionan características que permiten que un proceso contenga múltiples hilos de control.

- Un hilo es una unidad básica de utilización del CPU. Comprende un ID de hilo, un contador de programa, un conjunto de registros y una pila. Comparte con otros hilos que pertenecen al mismo proceso su sección de código, sección de datos y otros recursos del sistema operativo, como archivos abiertos y señales. El proceso tradicional (o pesado) tiene un solo hilo de control. Si un proceso tiene varios hilos de control, puede realizar más de una tarea a la vez.

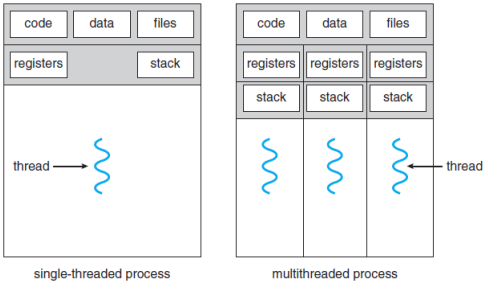

Figure 4.1 Single-threaded and multithreaded processes.

- La mayoría de las aplicaciones de software que se ejecutan en las computadoras modernas son multihilos. Por lo general, una aplicación se implementa como un proceso separado con varios hilos de control. Un navegador web puede tener un hilo que muestra imágenes o texto mientras que otro hilo recupera datos de la red, por ejemplo. Un procesador de texto puede tener un hilo para mostrar gráficos, otro hilo para responder a las pulsaciones de teclas del usuario y un tercer hilo para realizar la revisión ortográfica y gramatical en segundo plano. Las aplicaciones también se pueden diseñar para aprovechar las capacidades de procesamiento en sistemas multinúcleo. Estas aplicaciones pueden realizar varias tareas de uso intensivo del CPU en paralelo en los múltiples núcleos informáticos.

Beneficios

Los beneficios de la programación multihilo se pueden dividir en cuatro categorías principales:

- **Respuesta.** Una aplicación multihilos interactiva puede permitir que un programa continúe ejecutándose incluso si parte de él está bloqueado o está realizando una operación prolongada, aumentando así la capacidad de respuesta del usuario. Esta cualidad es especialmente útil en el diseño de interfaces de usuario.
- **Compartir recursos.** Los procesos solo pueden compartir recursos a través de técnicas como la memoria compartida y el paso de mensajes. Estas técnicas deben ser organizadas explícitamente por el programador. Sin embargo, los hilos comparten la memoria y los recursos del proceso al que pertenecen de forma predeterminada. El beneficio de compartir código y datos es que permite que una aplicación tenga varios hilos de actividad diferentes dentro del mismo espacio de direcciones.

- **Economía.** Asignar memoria y recursos para la creación de procesos es costoso. Debido a que los hilos comparten los recursos del proceso al que pertenecen, es más económico crear hilos y cambiar de contexto.
- **Escalabilidad.** Los beneficios del subproceso múltiple pueden ser aún mayores en una arquitectura de multiprocesador, donde los subprocesos pueden ejecutarse en paralelo en diferentes núcleos de procesamiento. Un proceso de un solo subproceso puede ejecutarse en un solo procesador, independientemente de cuántos estén disponibles.

- La programación multihilo proporciona un mecanismo para un uso más eficiente de estos múltiples núcleos informáticos y una mejor concurrencia. Considere una aplicación con cuatro hilos. En un sistema con un solo núcleo informático, la concurrencia simplemente significa que la ejecución de los hilos se intercalará con el tiempo (Figura 4.3), porque el núcleo de procesamiento es capaz de ejecutar solo un hilo a la vez. Sin embargo, en un sistema con múltiples núcleos, la concurrencia significa que los hilos pueden ejecutarse en paralelo, porque el sistema puede asignar un hilo separado a cada núcleo (Figura 4.4).

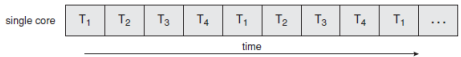

Figure 4.3 Concurrent execution on a single-core system.

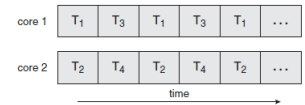

Figure 4.4 Parallel execution on a multicore system.

La tendencia hacia los sistemas multinúcleo sigue ejerciendo presión sobre los diseñadores de sistemas y los programadores de aplicaciones para hacer un mejor uso de los múltiples núcleos informáticos. En general, cinco áreas presentan desafíos en la programación de sistemas multinúcleo:

- **Identificación de tareas.** Esto implica examinar aplicaciones para encontrar áreas que se puedan dividir en tareas simultáneas separadas. Idealmente, las tareas son independientes entre sí y, por lo tanto, pueden ejecutarse en paralelo en núcleos individuales.

- **Equilibrio.** Al identificar las tareas que pueden ejecutarse en paralelo, los programadores también deben asegurarse de que las tareas realicen un trabajo igual de igual valor. En algunos casos, una determinada tarea puede no aportar tanto valor al proceso general como otras tareas. Es posible que el uso de un núcleo de ejecución independiente para ejecutar esa tarea no valga la pena.
- **División de datos.** Así como las aplicaciones se dividen en tareas separadas, los datos a los que acceden y manipulan las tareas deben dividirse para que se ejecuten en núcleos separados.
- **Dependencia de datos.** Los datos a los que acceden las tareas deben examinarse en busca de dependencias entre dos o más tareas. Cuando una tarea depende de los datos de otra, los programadores deben asegurarse de que la ejecución de las tareas esté sincronizada para adaptarse a la dependencia de los datos.

- **Prueba y depuración.** Cuando un programa se ejecuta en paralelo en varios núcleos, son posibles muchas rutas de ejecución diferentes. Probar y depurar dichos programas simultáneos es intrínsecamente más difícil que probar y depurar aplicaciones de un solo hilo.

- El **paralelismo de datos** se centra en distribuir subconjuntos de los mismos datos en varios núcleos informáticos y realizar la misma operación en cada núcleo. Considere, por ejemplo, sumar el contenido de un arreglo de tamaño N. En un sistema de un solo núcleo, un hilo simplemente sumaría los elementos [0]. . . [N - 1]. En un sistema de doble núcleo, sin embargo, el hilo A, que se ejecuta en el núcleo 0, podría sumar los elementos [0]. . . [N / 2 - 1] mientras que el hilo B, que se ejecuta en el núcleo 1, podría sumar los elementos [N / 2]. . . [N - 1]. Los dos hilos se ejecutarían en paralelo en núcleos informáticos separados.

- El **paralelismo de tareas** implica distribuir no datos sino tareas (hilos) a través de múltiples núcleos informáticos. Cada hilo está realizando una operación única. Diferentes hilos pueden estar operando con los mismos datos o pueden estar operando con diferentes datos. Considere nuevamente nuestro ejemplo anterior. En contraste con esa situación, un ejemplo de paralelismo de tareas podría involucrar dos hilos, cada uno de los cuales realiza una operación estadística única en el arreglo de elementos. Los hilos nuevamente operan en paralelo en núcleos de computación separados, pero cada uno está realizando una operación única.

Entonces, fundamentalmente, el paralelismo de datos implica la distribución de datos en múltiples núcleos y el paralelismo de tareas en la distribución de tareas en múltiples núcleos. En la práctica, sin embargo, pocas aplicaciones siguen estrictamente el paralelismo de datos o tareas. En la mayoría de los casos, las aplicaciones utilizan un híbrido de estas dos estrategias.