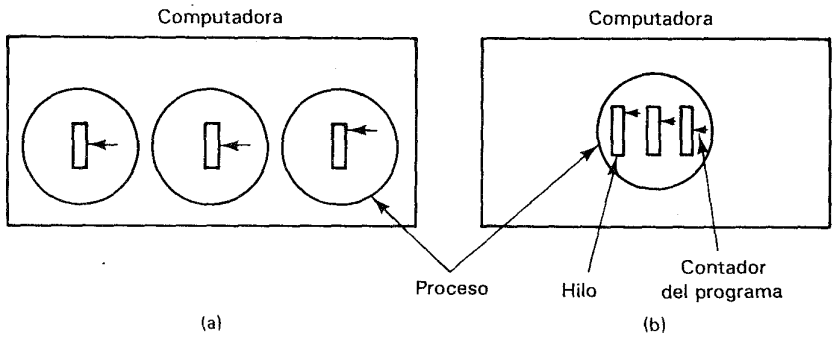
4.1. HILOS Y MULTIHILOS

En la mayoría de los sistemas operativos tradicionales, cada proceso tiene un espacio de direcciones y un hilo de control. De hecho, esta es casi la definición de un proceso. Sin embargo con frecuencia existen situaciones en donde se desea tener varios hilos de control que compartan un espacio de direcciones, pero que se ejecuten casi de manera cuasi-paralela como si fuesen de echo procesos independientes.

Consideremos, por ejemplo, un servidor de archivos que debe bloquearse en forma ocasional, en espera de acceso al disco. Si el servidor tiene varios hilos de control, se podría ejecutar un segundo hilo mientras el primero duerme. El resultado neto sería mejor rendimiento y desempeño. No es posible lograr este objetivo si se crean dos procesos servidores independientes, puesto que deben compartir un buffer caché común, lo que implica que deben estar en el mismo espacio de direcciones. Así, se necesita un nuevo mecanismo, uno que históricamente no se encontraba en los sistemas operativos de un procesador.



En la figura 4-1(a) vemos una máquina con tres procesos. Cada uno de ellos tiene su contador del programa, su pila, su conjunto de registros y su espacio de direcciones. Los procesos no tienen nada que ver entre sí, excepto que podrían comunicarse mediante las primitivas de comunicación entre procesos del sistema, como los semáforos, monitores o mensajes. En la figura 4-1(b) vemos otra máquina, con un proceso, sólo que ahora este proceso tiene varios hilos de control, los cuales por lo general se llaman sólo hilos o a veces procesos ligeros. En muchos sentidos, los hilos son como pequeños mini procesos. Cada hilo se ejecuta en forma estrictamente secuencial y tiene su contador de programa y una pila para llevar un registro de su posición. Los hilos comparten el CPU, de la misma forma que lo hacen los procesos primero, se ejecuta un hilo y después otro (tiempo compartido). Sólo en un multiprocesador se pueden ejecutar en realidad en paralelo. Los hilos pueden crear hilos hijos y se pueden bloquear en espera de que se terminen sus llamadas al sistema, al igual que los procesos regulares. Mientras un hilo está bloqueado, se puede ejecutar otro hilo del mismo proceso, en la misma forma en que, cuando se bloquea un proceso, se puede ejecutar en la misma máquina otro proceso. La analogía “hilo es a proceso como proceso es a máquina” es válida en muchos sentidos.

Sin embargo, los distintos hilos de un proceso no son tan independientes como los procesos distintos. Todos los hilos tienen el mismo espacio de direcciones, lo que quiere decir que comparten también las mismas variables globales. Puesto que cada hilo puede tener acceso a cada dirección virtual, un hilo puede leer, escribir o limpiar de manera completa la pila de otro hilo. No existe protección entre los hilos debido a que (1) es imposible y (2) no debe ser necesaria. A diferencia de los procesos distintos, que pueden ser de diversos usuarios y que pueden ser hostiles entre sí, un proceso siempre es poseído por un usuario, quien supuestamente ha creado varios hilos para que éstos cooperen y no luchen entre sí. Además de compartir un espacio de direcciones, todos los hilos comparten el mismo conjunto de archivos abiertos, procesos hijos, cronómetros, señales, etc. como se muestra en la figura 4-2. Así, la organización de la figura 4-1(a) se utilizaría en el caso en que los tres procesos en esencia no estuvieran relacionados, mientras que la figura 4-1(b) sería adecuada cuando los tres hilos fueran parte del mismo trabajo y cooperaran de manera activa y cercana entre sí.

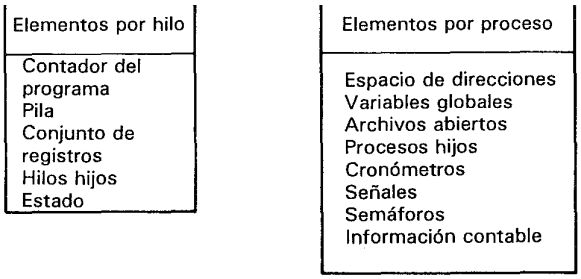


Figura 4-2. Conceptos por hilo y por proceso.

Como los procesos tradicionales (es decir, los procesos con un hilo), los hilos pueden tener uno de los siguientes estados: en ejecución, bloqueado, listo o terminado. Un hilo en ejecución posee CPU y está activo. Un hilo bloqueado espera que otro elimine el bloqueo (por ejemplo, en un semáforo). Un hilo listo está programado para su ejecución, la cual se llevará a cabo tan pronto le llegue su turno. Por último, un hilo terminado es aquel que ha hecho su salida, pero que todavía no es recogido por su padre (en términos de UNIX, el hilo padre no ha realizado un WAlT).

Uso de hilos

Los hilos se inventaron para permitir la combinación del paralelismo con la ejecución secuencial y el bloqueo de las llamadas al sistema. Consideremos de nuevo nuestro ejemplo del servidor de archivos. En la figura 4-3(a) se muestra una posible organización. En ésta, un hilo, el servidor, lee las solicitudes de trabajo del buzón del sistema. Después de examinar la solicitud elige a un hilo trabajador inactivo (es decir, bloqueado) y le envía la solicitud, lo cual se puede realizar al escribir un apuntador al mensaje, en una palabra especial asociada a cada hilo. El servidor despierta entonces al trabajador dormido.

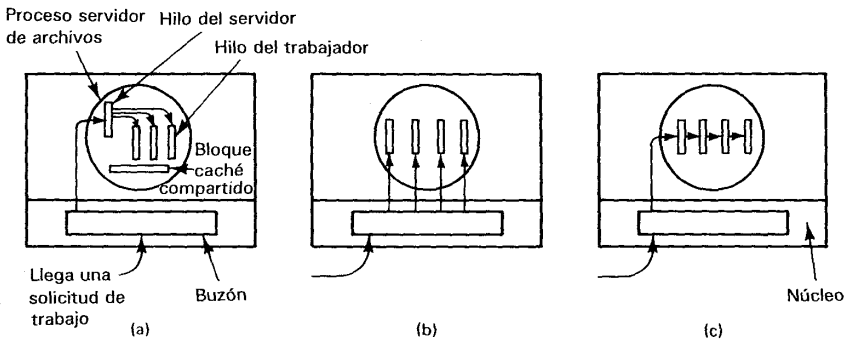


Figura 4-3. Tres organizaciones de hilos en un proceso. (a) Modelo del servidor/trabajador. (b) Modelo de equipo. (c) Modelo de entubamiento.

Cuando el trabajador despierta, verifica si puede satisfacer la solicitud por medio del bloque caché compartido, al que tienen acceso todos los hilos. Si no puede, envía un mensaje al disco para obtener el bloque necesario (si se trata de un READ) y se duerme en espera de la conclusión de la operación del disco. Se llama entonces al planificador y se inicia otro hilo, que tal vez sea el servidor, para pedir más trabajo; o bien, otro trabajador listo para realizar un trabajo.

Debe ser claro ahora lo que los hilos ofrecen. Ellos mantienen la idea de procesos secuenciales que hacen llamadas al sistema con bloqueo (por ejemplo, RPC para comunicarse al disco) y aun así logran un paralelismo. Este tipo de llamadas al sistema facilita la programación y el paralelismo mejora el desempeño. El servidor de un hilo mantiene el uso de estas llamadas al sistema, pero no aumenta el desempeño.

La estructura del servidor en la figura 4-3(a) no es la única manera de organizar un proceso de muchos hilos. El modelo de equipo de la figura 4-3(b) es también una posibilidad. Aquí todos los hilos son iguales y cada uno obtiene y procesa sus propias solicitudes. No hay servidor. A veces llega trabajo que un hilo no puede manejar, en particular si cada hilo se especializa en manejar cierto tipo de trabajo. En este caso, se puede utilizar una cola de trabajo, la cual contiene todos los trabajos pendientes. Con este tipo de organización, un hilo debe verificar primero la cola de trabajo antes de buscar en el buzón del sistema.

Los hilos también se pueden organizar mediante el modelo de entubamiento de la figura 4-3(c). En este modelo, el primer hilo genera ciertos datos y los transfiere al siguiente para su procesamiento. Los datos pasan de hilo en hilo y en cada etapa se lleva a cabo cierto procesamiento. Esta puede ser una buena opción en ciertos problemas, como el de los productores y los consumidores, aunque no es adecuado para servidores de archivos. Los entubamientos se utilizan con amplitud en muchas áreas de los sistemas de cómputo, desde la estructura interna de los Cpu RISC hasta las líneas de comandos de UNIX.

Con frecuencia, los hilos también son útiles para los clientes. Por ejemplo, si un cliente desea copiar un archivo en varios servidores, puede hacer que un hilo se comunique con cada servidor. Otro uso de los hilos clientes es el manejo de señales, como las interrupciones desde el teclado (DEL o BREAK). En vez de dejar que la señal interrumpa al proceso, se dedica un hilo de tiempo completo para la espera de señales.

Por lo general, éste se bloquea; pero cuando llega una señal, despierta y la procesa. Así, el uso de hilos puede eliminar la necesidad de las interrupciones a nivel usuario.

Otro argumento para los hilos no tiene que ver con RPC o la comunicación. Ciertas aplicaciones son más fáciles de programar mediante procesos paralelos; por ejemplo, el problema de los productores y los consumidores. El hecho de que el productor y el consumidor se ejecuten en paralelo es secundario. Se les programa de esa manera debido a que esto hace más simple el diseño del software. Como comparten un buffer común, no funcionaría el hecho de tenerlos en procesos ajenos. Los hilos se ajustan perfectamente a este caso.

Por último, aunque no estamos analizando este tema de manera explícita, en un sistema multiprocesador es posible que los hilos de un espacio de direcciones se ejecuten en realidad en paralelo, en varios CPU. De hecho, ésta es una de las formas principales para compartir los objetos en tales sistemas. Por otro lado, un programa diseñado de manera adecuada y que utilice hilos debe funcionar bien, tanto en un CPU con hilos compartidos, como en un verdadero multiprocesador, por lo que los aspectos del software son casi iguales de cualquier forma.