Principios de Sistemas Operativos

Hilos de ejecución (Threads)

Armando Arce, arce@itcr.ac.cr

Tecnológico de Costa Rica

Hilos de ejecución

Un hilo (o proceso ligero) es una unidad básica de utilización de la 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 la sección de código, la sección de datos y otros recursos de sistema operativo, como los archivos abiertos y las señales.
- Un proceso tradicional (o proceso pesado) tiene un solo hilo de control.
- Si un proceso tiene, por el contrario, múltiples hilos de control, puede realizar más de una tarea a la vez.

Hilos de ejecución

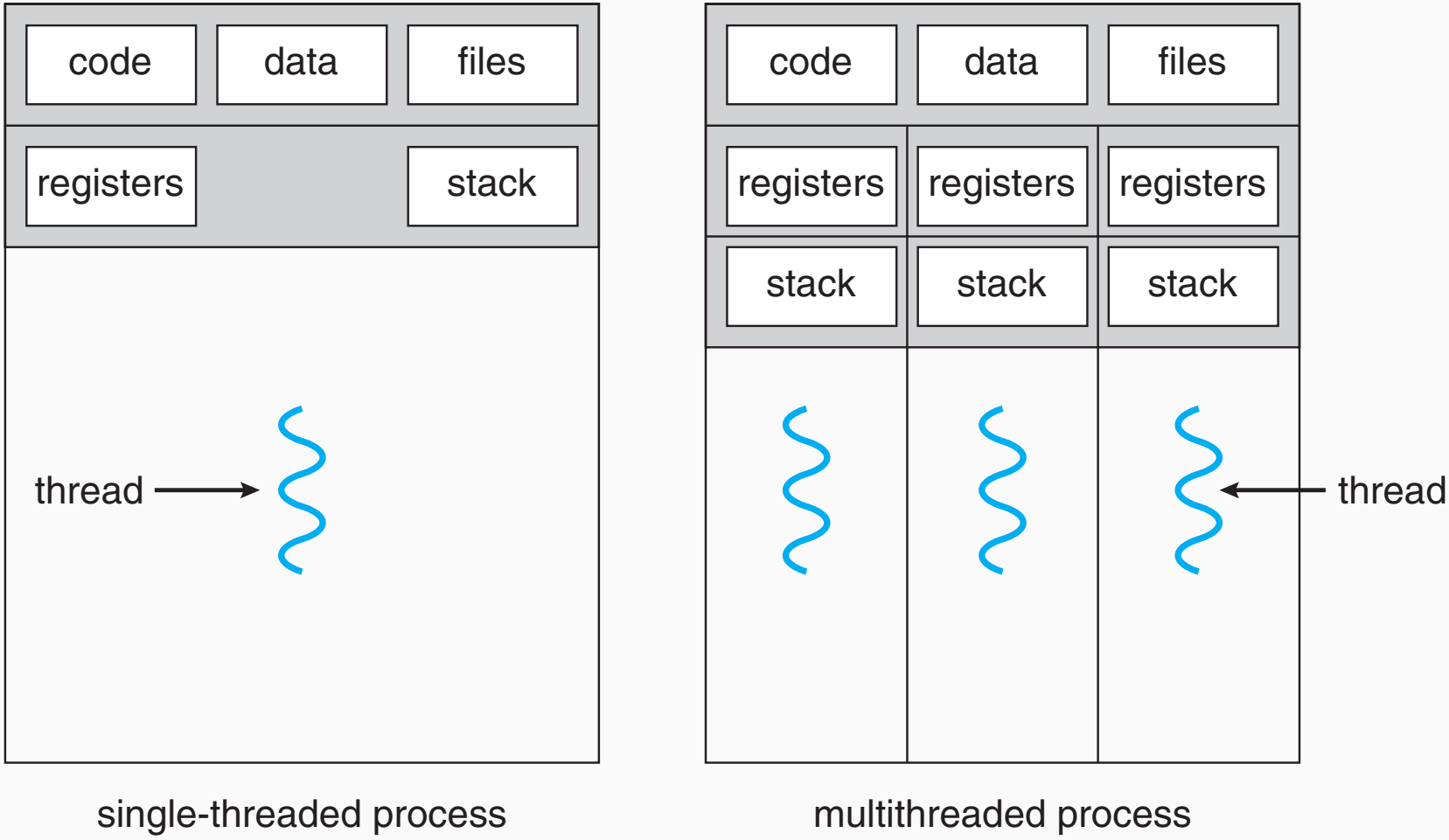

Figure 4.1 Single-threaded and multithreaded processes.

Figure 1:

Ventajas del uso de hilos

Las ventajas de la programación multihilos pueden dividirse en cuatro categorías principales:

- Capacidad de respuesta: el uso de múltiples hilos en una aplicación interactiva permite que un programa continúe ejecutándose incluso aunque parte de él esté bloqueado o realizando una operación muy larga, lo que incrementa la capacidad de respuesta al usuario.
- Compartición de recursos: por omisión, los hilos comparten la memoria y los recursos del proceso al que pertenece. La ventaja de compartir el código y los datos es que permite que una aplicación tenga varios hilos de actividad diferentes dentro del mismo espacio direcciones.

Ventajas del uso de hilos

- Economía: la asignación de memoria y recursos para la creación de procesos es costosa. Dado que los hilos comparten recursos del proceso al que pertenecen, es más económico crear y realizar cambio de contexto entre uno y otro hilo.
- Utilización sobre arquitecturas multiprocesador: las ventajas de usar configuraciones multihilos puede verse incrementada significativamente en una arquitectura multiprocesador, donde los hilos puede ejecutarse en paralelo en los diferentes pensadores.

Modelos multihilos (híbridos)

Desde el punto de vista práctico, el soporte para los hilos puede proporcionarse en el nivel de usuario (para los hilos de usuario) o por parte del kernel (para los hilos del kernel).

- El soporte para los hilos de usuarios se proporciona por encima del kernel y los hilos se gestionan sin soporte del mismo, mientras que el sistema operativo soporta y gestiona directamente los hilos del kernel.

Modelos multihilos (híbridos)

El alcance de contención proporciona al programador cierto control sobre la correspondencia entre las entidades de núcleo y los hilos.

- El usuario escribe el programa en términos de hilos a nivel de usuario y luego especifica cuántas entidades planificables por el núcleo serán asociadas al proceso:

Modelo muchos-a-uno

El modelo muchos-a-uno asigna múltiples hilos del nivel de usuario a un hilo del kernel.

- La administración de hilos se hace mediante la biblioteca en el espacio del usuario, por lo que resulta eficiente, pero el proceso completo se bloquea si un hilo realiza una llamada bloqueante al sistema.
- También, dado que sólo un hilo puede acceder al kernel a la vez, no podrán ejecutar varios hilos en paralelo sobre múltiples procesadores.

Modelo muchos-a-uno

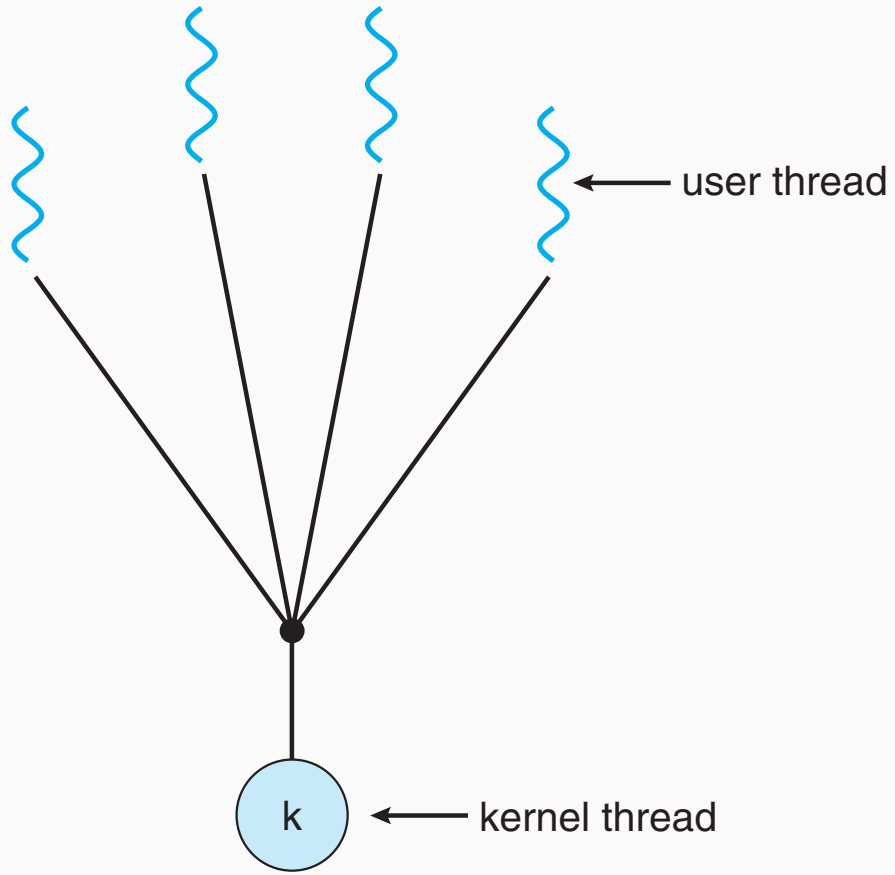

Figure 4.5 Many-to-one model.

Figure 2:

Modelo uno-a-uno

El modelo uno-a-uno asigna cada hilo de usuario a un hilo del kernel.

- Proporciona mayor concurrencia que el modelo muchos-a-uno, permitiendo que se ejecuten múltiples hilos en paralelo sobre varios procesadores.
- El único inconveniente de este modelo es que crear un hilo de usuario requiere crear el correspondiente hilo del kernel.
- Dado que la carga de trabajo administrativa para la creación de hilos del kernel puede repercutir en el rendimiento de una aplicación, la mayoría de las implementaciones de este modelo restringen el número de hilos soportados por el sistema.

Modelo uno-a-uno

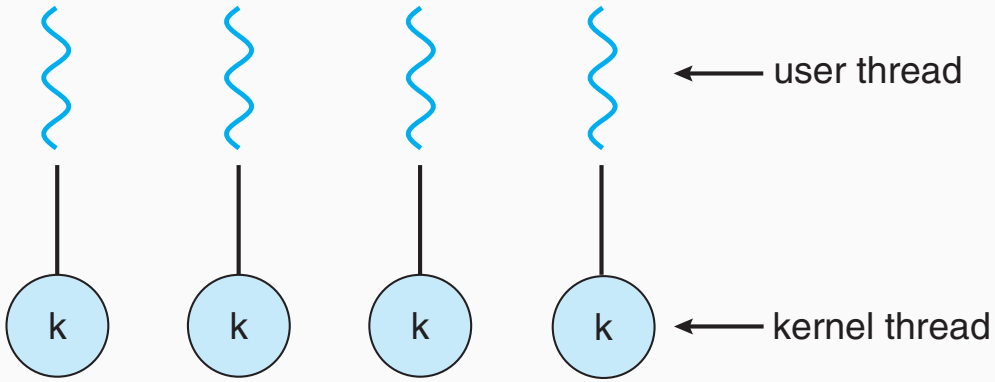

Figure 4.6 One-to-one model.

Figure 3:

Modelo muchos-a-muchos

El modelo muchos-a-muchos multiplexa muchos hilos de usuario sobre un número menor o igual de hilos del kernel.

- El número de hilos puede ser específico de una determinada aplicación o de una determinada máquina.
- Mientras que el modelo muchos-a-uno permite al desarrollador crear tantos hilos de usuario como desee, no se consigue una concurrencia real, ya que el kernel sólo puede planificar la ejecución de un hilo cada vez.

Modelo muchos-a-muchos

- El modelo uno-a-uno permite una mayor concurrencia, pero el desarrollador debe tener cuidado de no crear demasiados hilos dentro de la aplicación y en muchos casos, el número de hilos que puede crear está limitado.
- El modelo muchos-a-muchos no sufre ninguno de estos inconvenientes.
- Los desarrolladores pueden crear tantos hilos de usuario como sean necesarios y los correspondientes hilos del kernel pueden ejecutarse en paralelo en un multiprocesador.
- Asimismo, cuando un hilo realice una llamada bloquean al sistema, el kernel puede planificar otro hilo para su ejecución.

Modelo muchos-a-muchos

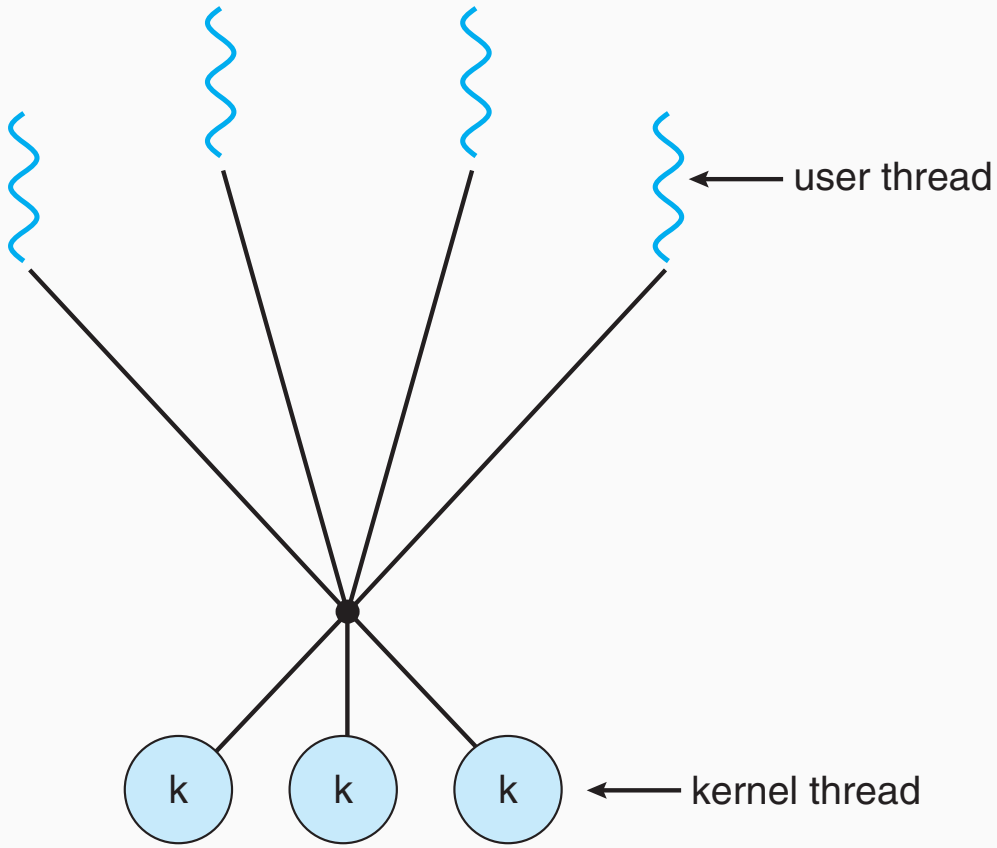

Figure 4.7 Many-to-many model.

Figure 4:

Modelo de dos niveles

Una variación del modelo muchos-a-muchos multiplexa muchos hilos del nivel de usuario sobre un número menor o igual de hilos del kernel, pero también permite acoplar un hilo de usuario a un hilo del kernel.

- A esta variante se le conoce como modelo de dos niveles.

Bibliotecas de hilos

Una biblioteca de hilos proporciona al programador un API para crear y gestionar hilos. Existen dos formas principales de implementar una biblioteca de hilos.

- El primer método consiste en proporcionar una biblioteca enteramente en el espacio del usuario, sin ningún soporte del kernel.
- Todas las estructuras de datos y el código de la biblioteca se encuentran en el espacio usuario.
- Esto significa que invocar a una función de la biblioteca es como realizar una llamada a una función local en el espacio del usuario y no una llamada al sistema.

Bibliotecas de hilos

El segundo método consiste en implementar una biblioteca en el nivel del kernel, soportada directamente por el sistema operativo.

- En este caso, el código y las estructuras de datos de la biblioteca se encuentran en el espacio del kernel.
- Invocar una función del API de la biblioteca normalmente da lugar a que se produzca una llamada al sistema dirigida al kernel.

Consideraciones sobre hilos

La programación de programas multihilos presenta una gran cantidad de retos, en esta sección se examinan algunos de ellos.

Cancelación de hilos

La cancelación de un hilo en la acción de terminarlo antes de que se haya completado.

- Por ejemplo, si varios hilos están realizando una búsqueda de forma concurrente en una base de datos y un hilo devuelve el resultado, los restantes hilos deben de ser cancelados.
- Puede producirse otra situación de este tipo cuando un usuario pulsa un botón en un explorador web que detiene la descarga de una página web.

Cancelación de hilos

A menudo, las páginas web se cargan usando varios hilos, cargándose cada imagen en un hilo diferente.

- Cuando un usuario pulsa el botón de cancelación en el navegador, todos los hilos de descarga de la página se cancelan.

Cancelación de hilos

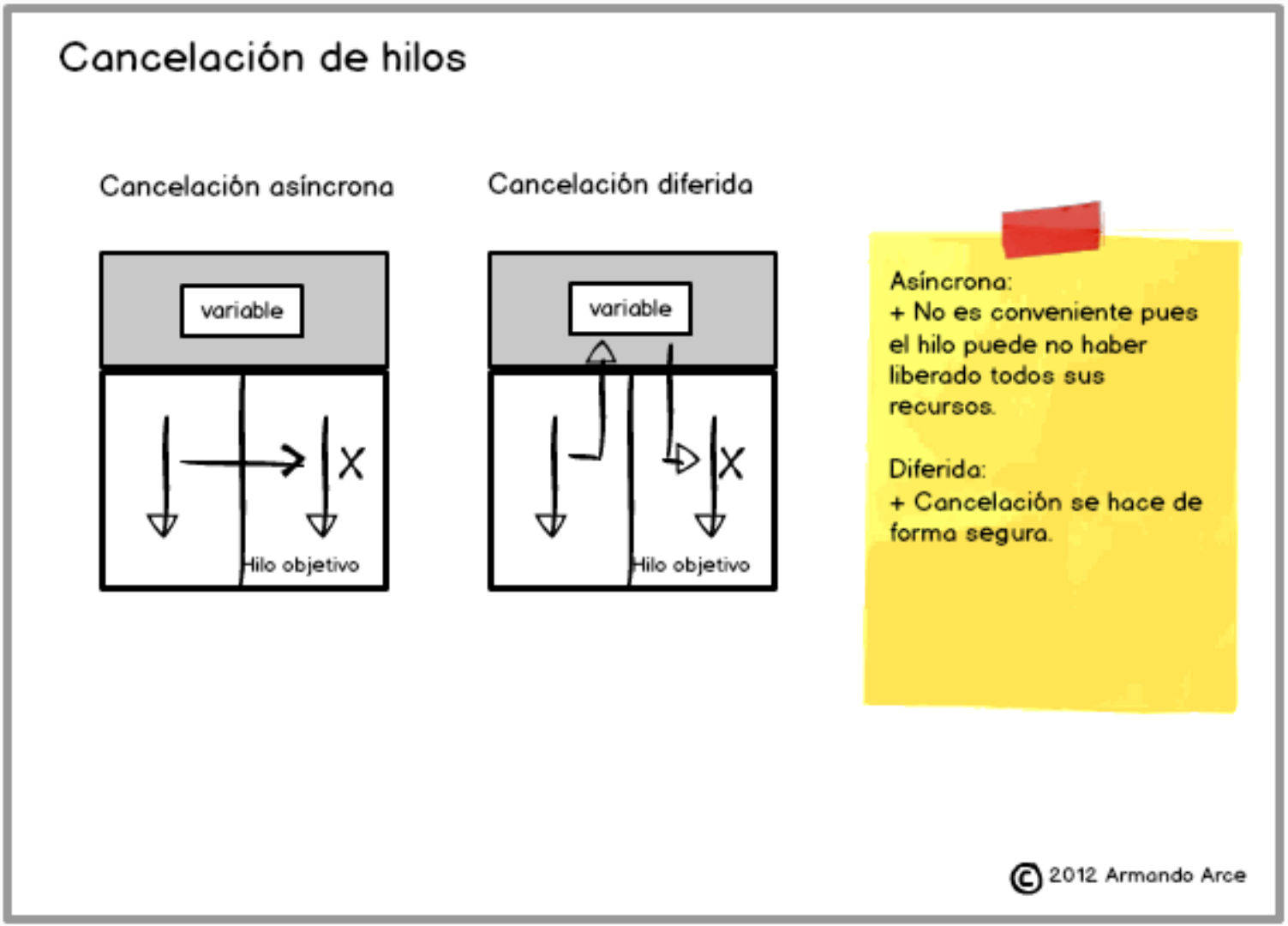

Figure 5:

Cancelación de hilos

El hilo que va ser cancelado se denomina a menudo hilo objetivo. La cancelación de un hilo objetivo puede ocurrir en dos escenarios diferentes:

- Cancelación asíncrona: un determinado hilo hace que termine inmediatamente el hilo objetivo.
- Cancelación diferida: el hilo objetivo comprueba periódicamente si debe terminar, lo que le proporciona una oportunidad de terminar por sí mismo de una manera ordenada.

Cancelación de hilos

La dificultad de la cancelación se produce en aquellas situaciones en las que se han asignado recursos al hilo cancelado o cuando un hilo se cancela en mitad de una actualización de datos que tuviera compartidos con otros hilos.

- Estos problemas son especialmente graves cuando se utiliza el mecanismo de cancelación asíncrona.
- A menudo, el sistema operativo reclamará los recursos asignados a un hilo cancelado, pero no reclamará todos los recursos.
- Por tanto, cancelar un hilo de forma asíncrona puede hacer que no se libere un recurso del sistema que sea necesario para otras cosas.

Cancelación de hilos

Por el contrario, con la cancelación diferida, un hilo indica que un hilo objetivo va a ser cancelado, pero la cancelación sólo se produce después que el hilo objetivo haya comprobado el valor de un indicador para determinar si debe cancelarse o no.

- Esto permite que esa comprobación de si debe cancelarse sea realizada por el hilo objetivo en algún punto en que la cancelación se pueda hacer de forma segura.

Tratamiento de señales

Una señal (o excepción) se usa para notificar a un proceso que se ha producido un determinado suceso.

- Una señal puede recibirse síncrona o asíncronamente, dependiendo del origen y de la razón por la que el suceso deba hacer señalizado.

Tratamiento de señales

Todas las señales, sean síncronas o asíncronas, siguen el mismo patrón:

1. Una señal se genera debido a que se produce un determinado suceso.
2. La señal generada se suministra a un proceso.
3. Una vez suministrada, la señal debe ser tratada.

Tratamiento de señales

Como ejemplo de señales síncronas se pueden citar los accesos ilegales a memoria y la división por 0.

- Si un programa en ejecución realiza una de estas acciones, se genera una señal.
- Las señales síncronas se proporcionan al mismo proceso que realizó la operación que causó la señal, esa es la razón por la que se consideran síncronas.

Tratamiento de señales

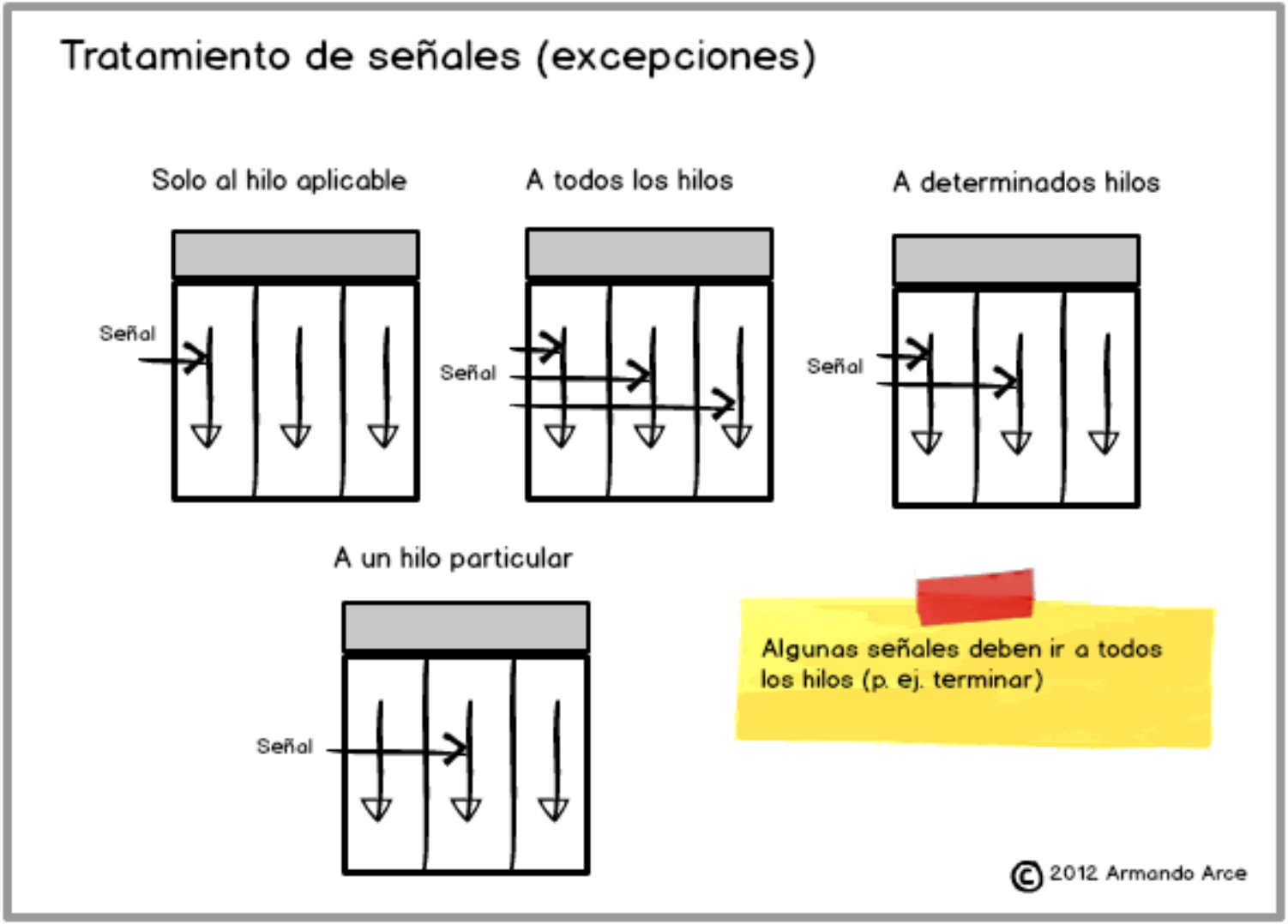

Figure 6:

Tratamiento de señales

Cuando una señal se genera por un suceso externo a un proceso de ejecución, dicho proceso recibe la señal en modo asíncrono.

- Como ejemplo de tales señales podemos citar la terminación de un proceso mediante la pulsación de teclas específicas, como , y el fin de cuenta de un temporizador.
- Normalmente, las señales asíncronas se envían a otro proceso.

Tratamiento de señales

Cada señal puede ser tratada por una de dos posibles rutinas de tratamiento:

- una rutina de tratamiento de señal predeterminada.
- una rutina de tratamiento de señal definida por el usuario.

Tratamiento de señales

Cada señal tiene una rutina de tratamiento de señal predeterminada que el kernel ejecuta cuando trata dicha señal.

- Esta acción predeterminada puede ser sustituida por una rutina de tratamiento de señal definida por el usuario a la que se llama para tratar la señal.
- Las señales pueden tratarse de diferentes formas: algunas señales (tales como la de cambio en el tamaño de una ventana) simplemente pueden ignorarse; otras (como un acceso ilegal la memoria) pueden tratarse mediante la terminación del programa.

Tratamiento de señales

El tratamiento de señales en programas de un solo hilo resulta sencillo; las señales siempre se suministran a un proceso.

- Sin embargo, suministrar las señales en los programas multihilos es más complicado, ya que un proceso puede tener varios hilos.

Tratamiento de señales

En general, hay disponibles las siguientes opciones:

1. Suministra la señal al hilo al que sea aplicable la señal.
2. Suministra la señal a todos los hilos del proceso.
3. Suministra la señal a determinados hilos del proceso.
4. Asignar un hilo específico para recibir todas las señales del proceso.

Tratamiento de señales

El método para suministrar una señal depende del tipo de señal generada.

- Por ejemplo, las señales síncronas tienen que suministrarse al hilo que causó la señal y no a los restantes hilos del proceso.
- Sin embargo, la situación con las señales asíncronas no está clara.
- Algunas señales asíncronas, como la señal de terminación de un proceso deberían enviarse a todos los hilos.

Conjuntos compartidos de hilos

Cuando un servidor web recibe una solicitud, crea un hilo nuevo para dar servicio a la solicitud.

- Aunque crear un nuevo hilo es, indudablemente, mucho menos costoso que crear un proceso nuevo, los servidores multihilo no están exentos de problemas potenciales.
- El primero concierne a la cantidad de tiempo necesario para crear el hilo antes de dar servicio a la solicitud, junto con el hecho de que este hilo se descartará una vez que haya completado su trabajo.

Conjuntos compartidos de hilos

- El segundo tema es más problemático: si se permite que todas las solicitudes concurrentes sean servidas mediante un nuevo hilo, no se puede limitar el número de hilos activos de forma concurrente en el sistema.
- Un número ilimitado de hilos podría agotar determinados recursos del sistema, como el tiempo de CPU o la memoria.
- Una solución para este problema consiste en usar lo que se denomina un conjunto compartido de hilos.

Conjuntos compartidos de hilos

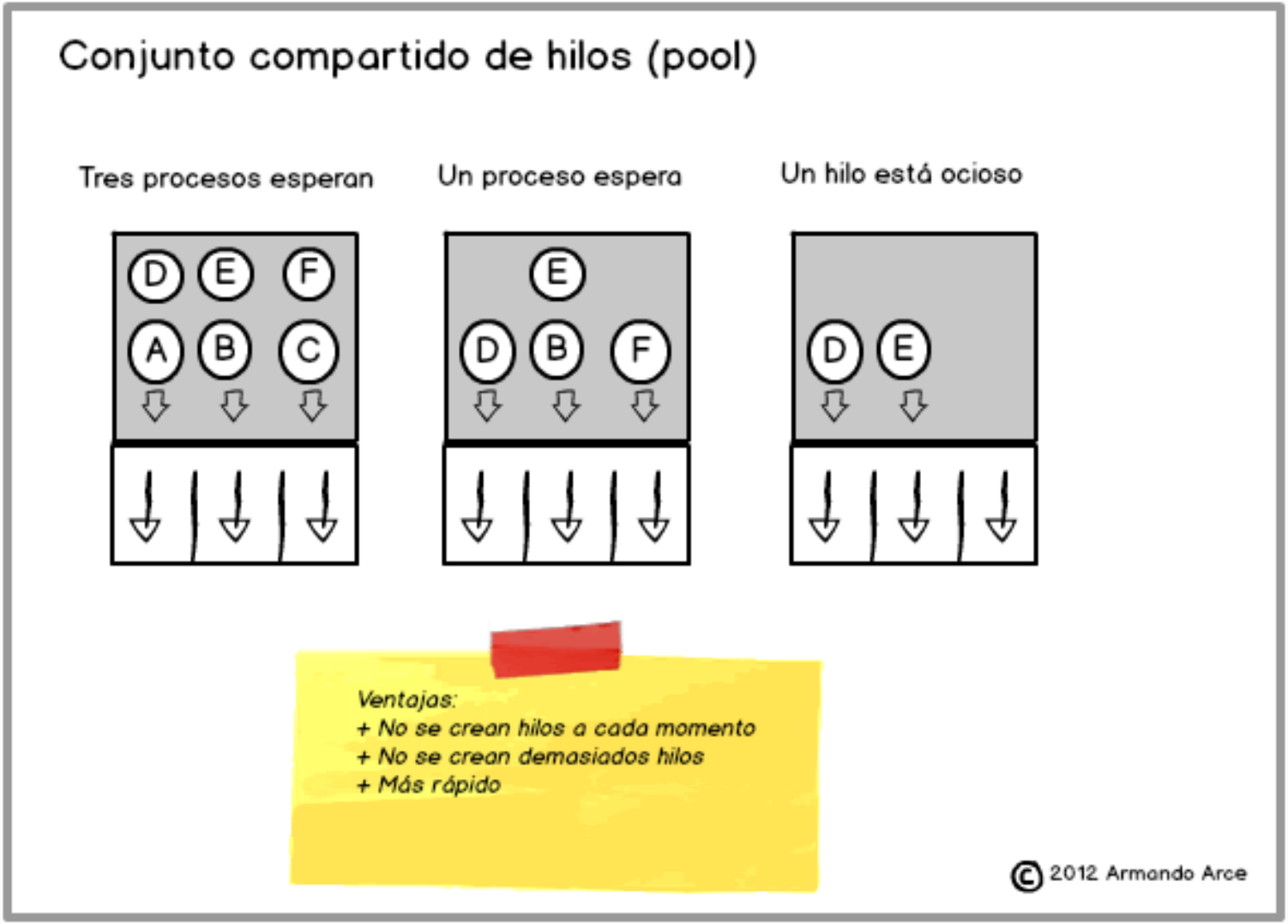

Figure 7:

Conjuntos compartidos de hilos

La idea general subyacente a los conjuntos de hilos en la de crear una serie de hilos al principio del proceso y colocarlos en un conjunto compartido, donde los hilos queden a la espera de que se les asigne un trabajo.

- Cuando un servidor recibe una solicitud, despierta a un hilo del conjunto, si hay uno disponible, y le pasa la solicitud para que el hilo se encargue de darle servicio.
- Una vez que el hilo completa su tarea, vuelve al conjunto compartido y espera hasta que haya más trabajo.
- Si el conjunto no tiene ningún hilo disponible, el servidor espera hasta que quede uno libre.

Conjuntos compartidos de hilos

Los conjuntos compartidos de hilos ofrece las siguientes ventajas:

- Dar servicio a una solicitud con un hilo existente normalmente es más rápido que esperar a crear un nuevo hilo
- Un conjunto de hilos limita el número de hilos existentes en cualquier instante dado. Esto es particularmente importante en aquellos sistemas que no pueden soportar un gran número de hilos concurrentes.

Conjuntos compartidos de hilos

El número de hilos del conjunto pueden definirse heurísticamente basándose en factores como el número de procesadores del sistema, la cantidad de memoria física y el número esperado de solicitudes concurrentes de los clientes.

- Las arquitecturas de conjuntos compartidos más complejas pueden ajustar dinámicamente el número de hilos del conjunto de acuerdo con los patrones de uso.
- Estas arquitecturas proporcionan la importante ventaja de tener un conjunto compartido más pequeño, que consume menos memoria, cuando la carga el sistema es baja.

Activación del planificador

En este caso el núcleo asigna un cierto número de procesadores virtuales a cada proceso y deja que el sistema en tiempo de ejecución (en el espacio de usuario) asigne hilos a procesadores.

- El número de procesadores virtuales asignados a un proceso es inicialmente de uno sólo, pero el proceso puede pedir más y puede también devolver procesadores que ya no necesite.

Activación del planificador

Cuando el núcleo detecta que un hilo se ha bloqueado, el núcleo se lo notifica al sistema de tiempo de ejecución del proceso, pasándole como parámetros sobre la pila el número de hilo en cuestión y una descripción del suceso que ha tenido lugar. Esto se realiza mediante una retrollamada.

- De esta forma el sistema sabe que un hilo se ha bloqueado y planifica otros hilos.

Activación del planificador

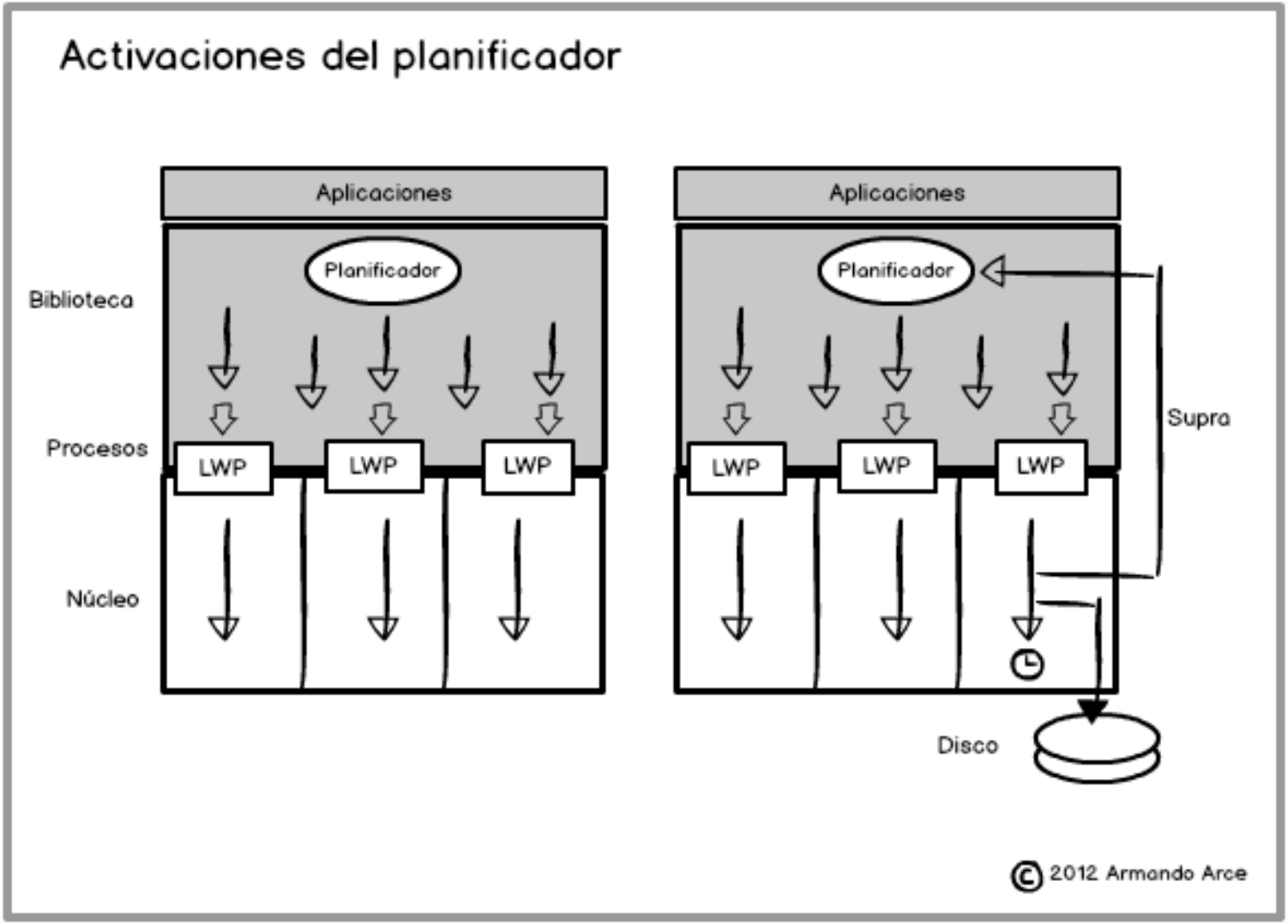

Figure 8:

Activación del planificador

Muchos sistemas que implementan el modelo muchos-a-muchos o el modelo de dos niveles colocan una estructura de datos intermedia entre los hilos de usuario y del kernel.

- Esta estructura de datos es conocida como proceso ligero (LWP). A la biblioteca de hilos de usuario, el proceso ligero le parece un procesador virtual en el que la aplicación puede hacer que se ejecute un hilo de usuario.
- Cada LWP se asocia a un hilo del kernel, y es este hilo del kernel el que el sistema ejecuta en los procesadores físicos.

Activación del planificador

La activación del planificador funciona como sigue:

- El kernel proporciona a la aplicación un conjunto de procesadores virtuales (LWP) y la aplicación puede planificar la ejecución de los hilos de usuario en uno de los LWP disponibles.
- Además, el kernel debe informar a la aplicación sobre determinados sucesos; este procedimiento se conoce con el nombre de suprallamada (upcall).

Activación del planificador

Las suprallamadas son tratadas por la biblioteca de hilos mediante una rutina de tratamiento de suprallamadas, y estas rutinas deben ejecutarse sobre un procesador virtual.

- Uno de los sucesos que disparan una suprallamada se produce cuando un hilo de la aplicación esté a punto de bloquearse.
- En este escenario, el kernel realiza una suprallamada a la aplicación para informarla de que un hilo se va a bloquear y para identificar el hilo concreto de que se trata.
- El kernel asigna entonces un LWP nuevo a la aplicación.

Activación del planificador

- La aplicación ejecuta la rutina de tratamiento de la suprallamada sobre el nuevo LWP, que guarda el estado del hilo bloqueante, y libera el LWP en el que se está ejecutando dicho hilo.
- La rutina de tratamiento de la suprallamada planifica entonces la ejecución de otro hilo que reúna los requisitos para ejecutarse en el nuevo LWP.
- Cuando se produce el suceso que el hilo bloqueante estaba esperando, el kernel hace otra suprallamada a la biblioteca de hilos para informar que el hilo anteriormente bloqueado ahora puede ejecutarse.

Referencia bibliográfica

A.SILBERSCHATZ, P. GALVIN, y G. GAGNE, Operating Systems Concepts, Cap.4, 9a Edición, John Wiley, 2013.