Tema

3.1

Sistema Operativos (SSOO)

Procesos e Hilos

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Índice

- Procesos e hilos.
- Multiprocesamiento simétrico.
- Micronúcleos.

Procesos e hilos

- En los SO tradicionales, cada proceso tiene un espacio de direcciones y un solo hilo de control. De hecho, ésa es casi la definición de un proceso.
- Un proceso es algo más complejo de lo que se ha visto hasta este momento.
- Contiene dos conceptos diferentes y potencialmente independientes:
  - uno relativo a la propiedad de recursos y
  - Otro relativo a la ejecución.
- Esta distinción ha llevado al desarrollo de los hilos o *threads*.

Procesos e hilos

Características de un proceso:

1. **Recursos:** un proceso incluye un espacio de direcciones virtuales para el manejo de la imagen del proceso.
   - La imagen de un proceso es la colección de programa, datos, pila y atributos definidos en el bloque de control del proceso.
   - A un proceso se le puede asignar control o propiedad de recursos tales como la memoria principal, canales E/S, dispositivos E/S y archivos.
   - El sistema operativo se encarga de la protección para evitar interferencias no deseadas entre procesos en relación con los recursos.

Procesos e hilos

Características de un proceso:

2. **Planificación y ejecución:** la ejecución de un proceso sigue una ruta de ejecución (traza) a través de uno o más programas.
   - La ejecución puede estar intercalada con ese u otros procesos.
   - Un proceso tiene un estado de ejecución: Ejecutando, Listo, etc..
   - Tiene asignada una prioridad de activación, siendo ésta la entidad que planifica y activa el sistema operativo.

Procesos e hilos

Estas dos características son independientes y podrían ser tratadas como tales por el sistema operativo.

Para distinguir estas dos características, la unidad que se activa se suele denominar **hilo (thread)**, o **proceso ligero**, mientras que la unidad de propiedad de recursos se suele denominar **proceso** o **tarea**.

Procesos e hilos

¿Por qué definir hilos?

- Muchas aplicaciones realizan varias actividades a la vez.
- Algunas de ésas se pueden bloquear de vez en cuando.
- Al descomponer una aplicación en varios hilos secuenciales que se ejecutan en cuasi-paralelo, el modelo de programación se simplifica.

¿No era lo mismo para los procesos?

- Sí, se pensaba en procesos paralelos.
- Los hilos añaden un nuevo elemento: la habilidad de las entidades en paralelo de **compartir un espacio de direcciones y todos sus datos entre ellas**.
- Esta habilidad es esencial para ciertas aplicaciones, razón por la cual no funcionará el tener varios procesos (con sus espacios de direcciones separados).

Procesos e hilos

¿Por qué definir hilos?

- Los hilos son más ligeros que los procesos.
- Más fáciles de crear y de destruir *(de 10 a 100 veces más rápido)*.
- Los hilos no producen aumento del rendimiento cuando todos están ligados en la CPU.
- Cuando hay una gran cantidad de cálculos y operaciones de E/S, el uso de hilos permite solapar con lo cual agiliza la velocidad de la aplicación.
- Los hilos son útiles en los sistemas con varias CPU, donde si es posible un verdadero paralelismo.

Multihilo

Capacidad de un sistema operativo de dar soporte a múltiples hilos de ejecución en un solo proceso.

El enfoque de un solo hilo de ejecución por proceso, en el que no se identifica con el concepto de hilo, se conoce como estrategia monohilo.

- MS-DOS solo soporta un único proceso y un único hilo.
- Java soporta un único proceso con múltiples hilos.
- Windows, Solaris, Os soporta múltiples procesos y múltiples hilos.

Multihilo

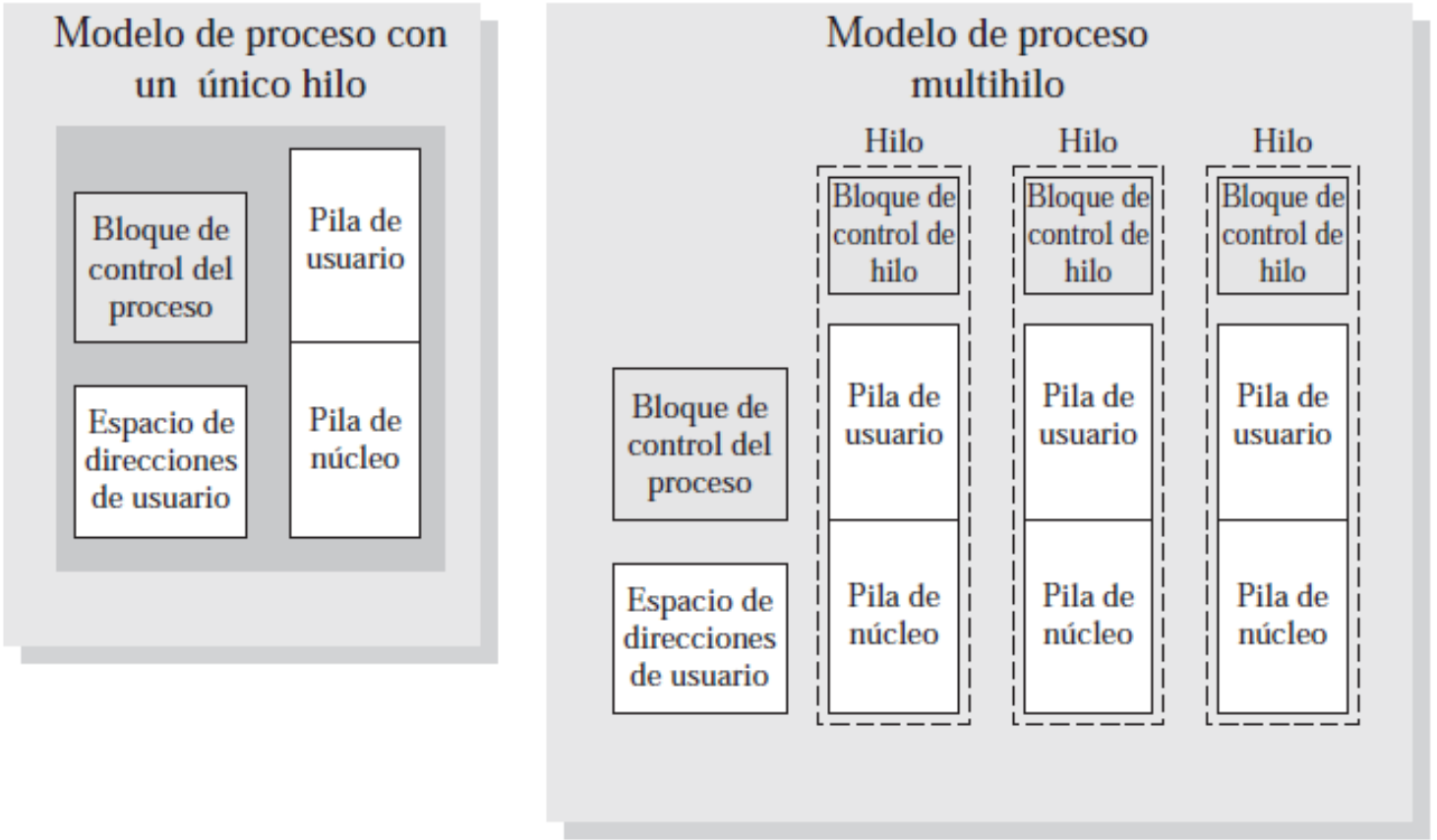


Figura 4.2. Modelos de proceso con un único hilo y multihilo.

Multihilo

Ejemplos de uso de un sistema Multihilo.

- Trabajo en primer plano y en segundo plano:
  - Esta forma de trabajo a menudo incrementa la velocidad que se percibe de la aplicación, permitiendo al programa solicitar el siguiente mandato antes de que el mandato anterior esté completado.
- Procesamiento asíncrono:
  - Los elementos asíncronos de un programa se pueden implementar como hilos.
- Velocidad de ejecución:
  - Un proceso Multihilo puede computar una serie de datos mientras que lee los siguientes de un dispositivo. Aunque un hilo pueda estar bloqueado por una operación de E/S mientras lee datos, otro hilo se puede estar ejecutando.

Multihilo

Ejemplos de uso de un sistema Multihilo.

- **Estructura modular de programas:**
  - Los programas que realizan diversas tareas o que tienen varias fuentes y destinos de entrada y salida, se pueden diseñar e implementar más fácilmente usando hilos.

Multihilo

La planificación y la activación se realizan a nivel de hilo; por ello la mayor parte de la información de estado relativa a su ejecución se mantiene en estructuras de datos que están a su nivel.

Elementos por proceso	Elementos por hilo
Espacio de direcciones	Contador de programa
Variables globales	Registros
Archivos abiertos	Pila
Procesos hijos	Estado
Alarmas pendientes	
Señales y manejadores de señales	
Información contable	

Multihilo – Estado de los hilos

- Los principales estados de los hilos son: Ejecutando, Listo y Bloqueado.
- No tiene sentido usar estados de suspensión a un hilo porque este es estado aplicado a nivel de proceso.

¿Si se expulsa un proceso, todos sus hilos se deben expulsar porque comparten el espacio de direcciones del proceso?

Las cuatro operaciones asociadas a los hilos son:

- **Creación:** con un nuevo proceso se crea un hilo. Un hilo puede luego crear otro hilo dentro del mismo proceso, facilitando argumentos e instrucciones al nuevo hilo.
- **Bloqueo:** si un hilo necesita esperar por un evento se bloquea (salva registros de usuario, el CP y punteros de pila). El procesador ejecuta otro hilo en estado **Listo**, dentro del **mismo proceso o en otro diferente.**
- **Desbloqueo:** cuando ocurre el evento por el que se ha bloqueado el hilo
- **Finalización:** al completar el hilo (se libera su registro y pilas)

Multihilo – Sincronización de hilos

- Los hilos de un proceso comparten el mismo espacio de direcciones y otros recursos.
- Cualquier alteración de un recurso por cualquiera de los hilos, afecta al entorno del resto de los hilos del mismo proceso.
- Es necesario sincronizar las actividades de los hilos para que no interfieran entre ellos o corrompan estructuras de datos.
  - Imagine el caso en que dos hilos intentan añadir simultáneamente, un elemento a una lista doblemente enlazada. Podría perderse un elemento o la lista podría acabar malformada.
- **POSIX:** IEEE definió un estándar para los hilos conocido como 1003.1c. Este paquete se conoce como **Pthreads** y la mayoría de los sistemas UNIX aceptan dicho paquete.
  - El estándar define más de 60 llamadas a funciones, que son demasiadas como para verlas en este libro.

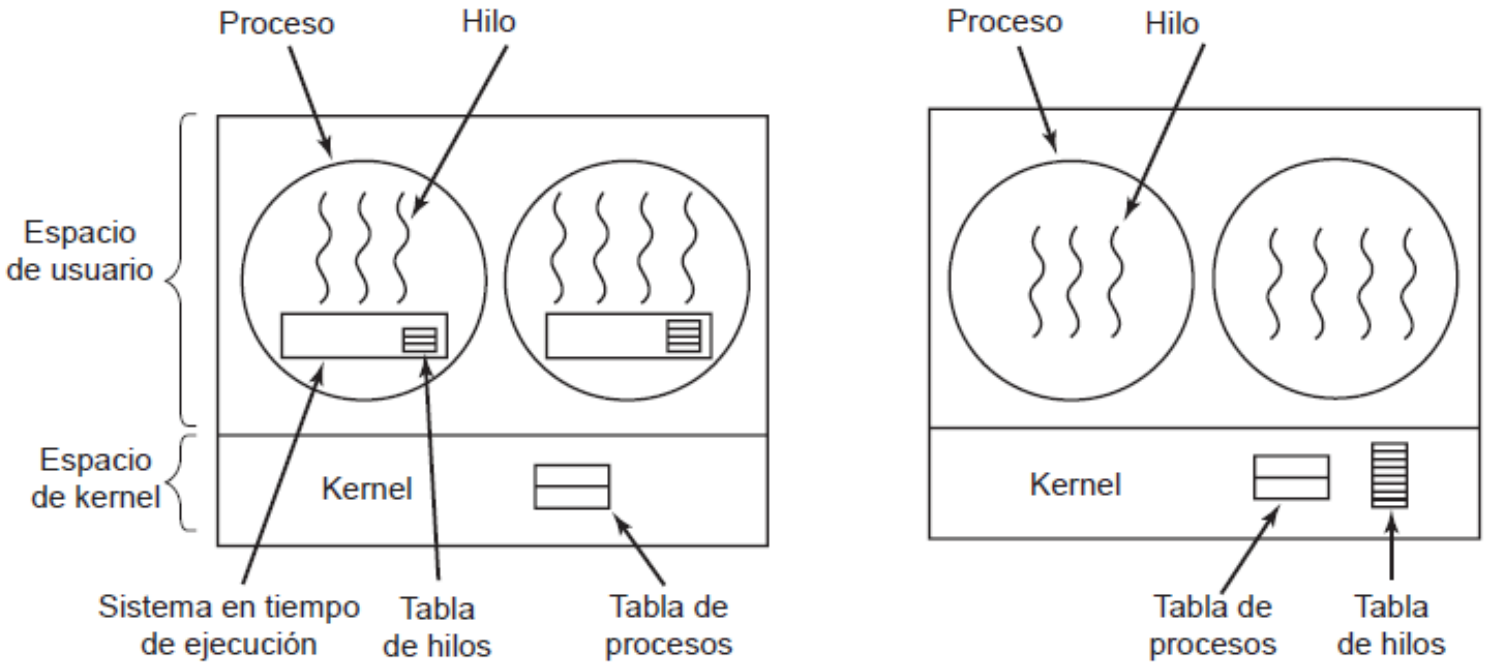
Multihilo – POSIX

- IEEE definió un estándar para los hilos conocido como 1003.1c.
- Este paquete se conoce como **Pthreads.**
- La mayoría de los sistemas UNIX aceptan dicho paquete.
- El estándar define más de 60 llamadas a funciones, que son demasiadas como para verlas en este libro.

Llamada de hilo	Descripción
Pthread_create	Crea un nuevo hilo
Pthread_exit	Termina el hilo llamador
Pthread_join	Espera a que un hilo específico termine
Pthread_yield	Libera la CPU para dejar que otro hilo se ejecute
Pthread_attr_init	Crea e inicializa la estructura de atributos de un hilo
Pthread_attr_destroy	Elimina la estructura de atributos de un hilo

Tipos de implementación de hilos

- **Hilos en espacio de usuario ULT**: consiste en ubicar los hilos en espacio de usuario. En este caso el kernel no conoce la existencia de dichos hilos, este cree que administra procesos con un solo hilo.



Tipos de implementación de hilos

Hilos en espacio de usuario ULT

- **Ventajas**
  - Un paquete de hilos de nivel usuario puede implementarse en un sistema operativo que no acepte hilos.
  - Cada proceso tenga su propio algoritmo de planificación personalizado.
  - El procedimiento que guarda el estado del hilo y el planificador son sólo procedimientos locales, por lo que es mucho más eficiente invocarlos que realizar una llamada al kernel (la planificación de hilos es muy rápida).
- **Problemas**
  - La manera en que se implementan las llamadas al sistema de bloqueo.
  - Los hilos pueden utilizar llamadas de bloqueo para evitar que un hilo bloqueado afectara a los demás, pero esto suele generar muchos problemas.
  - Cambiar esta implementación requiere modificar muchas aplicaciones de usuario.
  - Se usa el sistema de **envoltura** para comprobar posibles bloqueos *(es torpe)*.

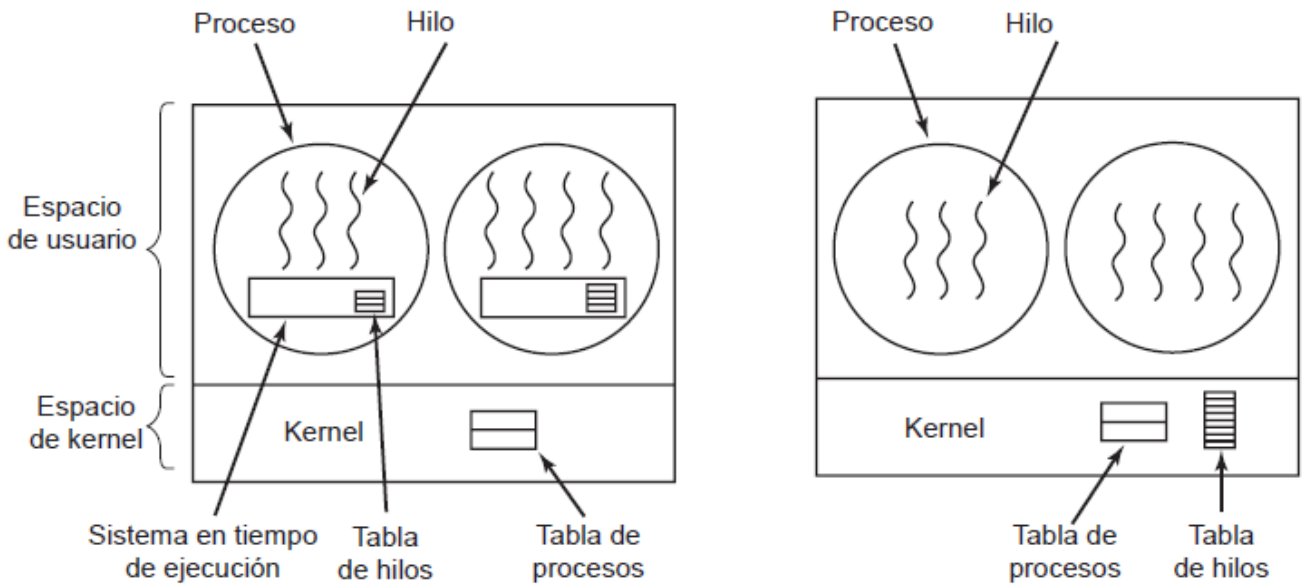
Tipos de implementación de hilos

Hilos en espacio de usuario ULT

- **Problemas**
  - Fallos de página: el ordenador se pueden configurar para que no todo el programa se encuentre en memoria a la vez. Si el programa llama o salta a una instrucción que no esté en memoria, ocurre un fallo de página y el sistema operativo obtiene la instrucción faltante (y las instrucciones aledañas) del disco.
  - El proceso se bloquea mientras la instrucción necesaria se localiza y se lee.
  - Si un hilo produce un fallo de página, el kernel (quien no sabe de la existencia de los hilos) bloquea naturalmente **todo el proceso** hasta que se complete la operación de E/S, incluso si otros hilos pudieran ser ejecutados.

Tipos de implementación de hilos

- **Hilos en el kernel KLT**: el kernel tiene una tabla de hilos que lleva la cuenta de todos los hilos en el sistema. Cuando un hilo desea crear un nuevo hilo o destruir uno existente, realiza una llamada al kernel, la cual se encarga de la creación o destrucción mediante una actualización en la tabla de hilos del kernel.



Tipos de implementación de hilos

Hilos en el kernel KLT: características

- Cuando un hilo se bloquea, el kernel puede ejecutar otro hilo del mismo proceso (si hay otro listo) o un hilo de un proceso distinto.
- Los hilos de kernel no requieren de nuevas llamadas al sistema sin bloqueo.
- Si un hilo en un proceso produce un fallo de página, el kernel puede comprobar con facilidad si el proceso tiene otros hilos que puedan ejecutarse y de ser así, ejecuta uno de ellos mientras espera a que se traiga la página requerida desde el disco.

- **Problemas**
  - El costo de una llamada al sistema es considerable, por lo que si las operaciones de hilos son comunes, habrá sobrecarga.
  - El uso de *fork*() para crear otro proceso con muchos hilos
  - Las señales se envían a los procesos, no a los hilos (modelo clásico).
  - Los hilos de kernel son mejores que los hilos de usuario, pero estos son más lentos.

Recordar: Interrupción

Interrupciones:

- Señal que envía un dispositivo de E/S a la CPU para indicar que la operación de la que estaba ocupado, ya ha terminado. Las interrupciones son manejadas por el procesador después de que finaliza la instrucción actual.
- **Interrupciones por Hardware:** el hardware avisa al SO cuando el dispositivo de E/S ha terminado para que este intervenga y hacer que el programa que estaba esperando el dispositivo, continúe su ejecución.
- **Interrupciones por Software**: se produce cuando un usuario solicita una llamada del sistema (a través de un programa).
- Ambas permiten al SO utilizar la CPU en otras aplicación, mientras otra permanece a la espera de que concluya una operación en un dispositivo de E/S.

- **Interrupciones TRAP**:
  - Estas instrucciones permiten que un programa genere una interrupción. Estas instrucciones se emplean fundamentalmente para solicitar los servicios del sistema operativo.

Recordar: Interrupción

Interrupciones TRAP:

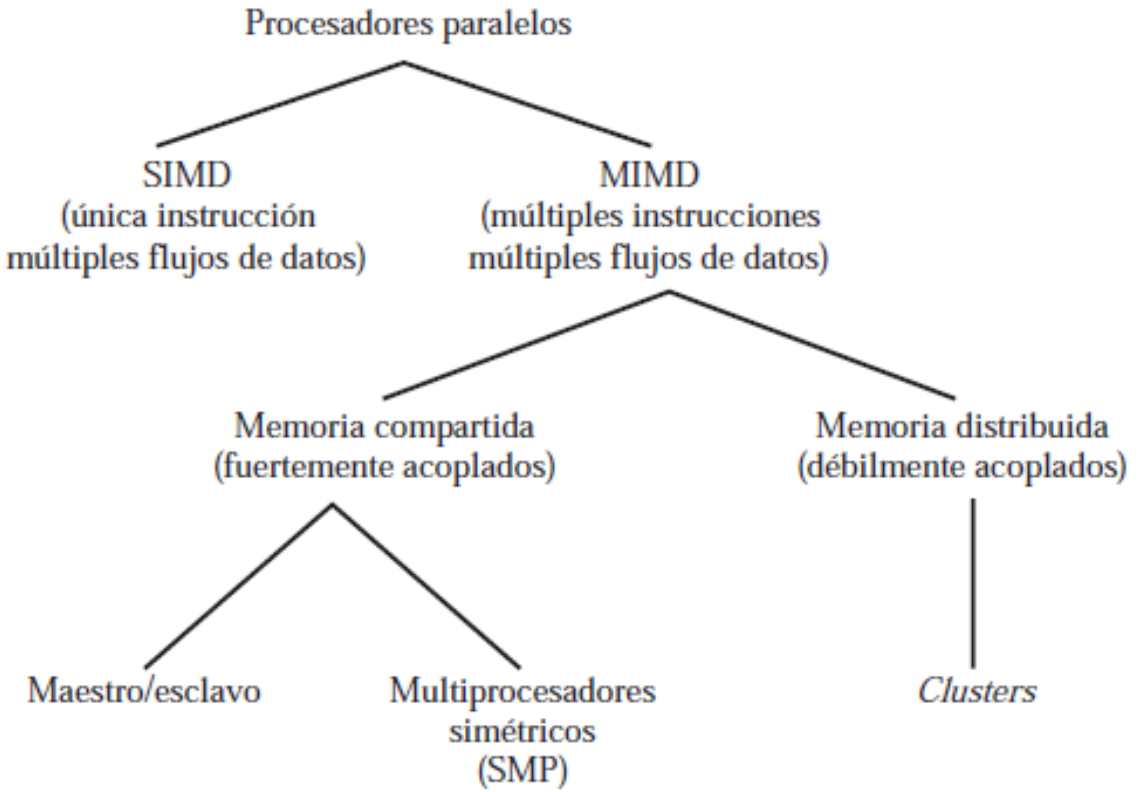
- Estas instrucciones permiten que un programa genere una interrupción. Estas instrucciones se emplean fundamentalmente para solicitar los servicios del SO.
- La forma en que se realiza una llamada al sistema consiste en colocar un conjunto de parámetros en un lugar específico, para después ejecutar una instrucción del lenguaje máquina del procesador denominada **trap** (en castellano, trampa).
- La ejecución de esta instrucción máquina hace que el hardware guarde el contador de programa y la palabra de estado del procesador (PSW, Processor Status Word) en un lugar seguro de la memoria, cargándose un nuevo contador de programa y una nueva PSW.
- Este nuevo contador de programa contiene una dirección de memoria donde reside una parte (un programa) del SO, el cual se encarga de llevar a cabo el servicio solicitado. Cuando el sistema operativo finaliza el servicio, coloca un código de estado en un registro para indicar si hubo éxito o fracaso, y ejecuta una instrucción ***return from trap***, esta instrucción provoca que el hardware restituya el contador de programa y la PSW del programa que realizó la llamada al sistema, prosiguiéndose así su ejecución.

Multiprocesamiento simétrico: SMP

Los dos enfoques más populares para proporcionar paralelismo a través de la réplica de procesadores: **multiprocesamiento simétricos (SMP)** y clústers.

La arquitectura SMP puede encajar dentro de las categorías de procesamiento paralelo.

- SISD
- SIMD
- MISD
- MIMD



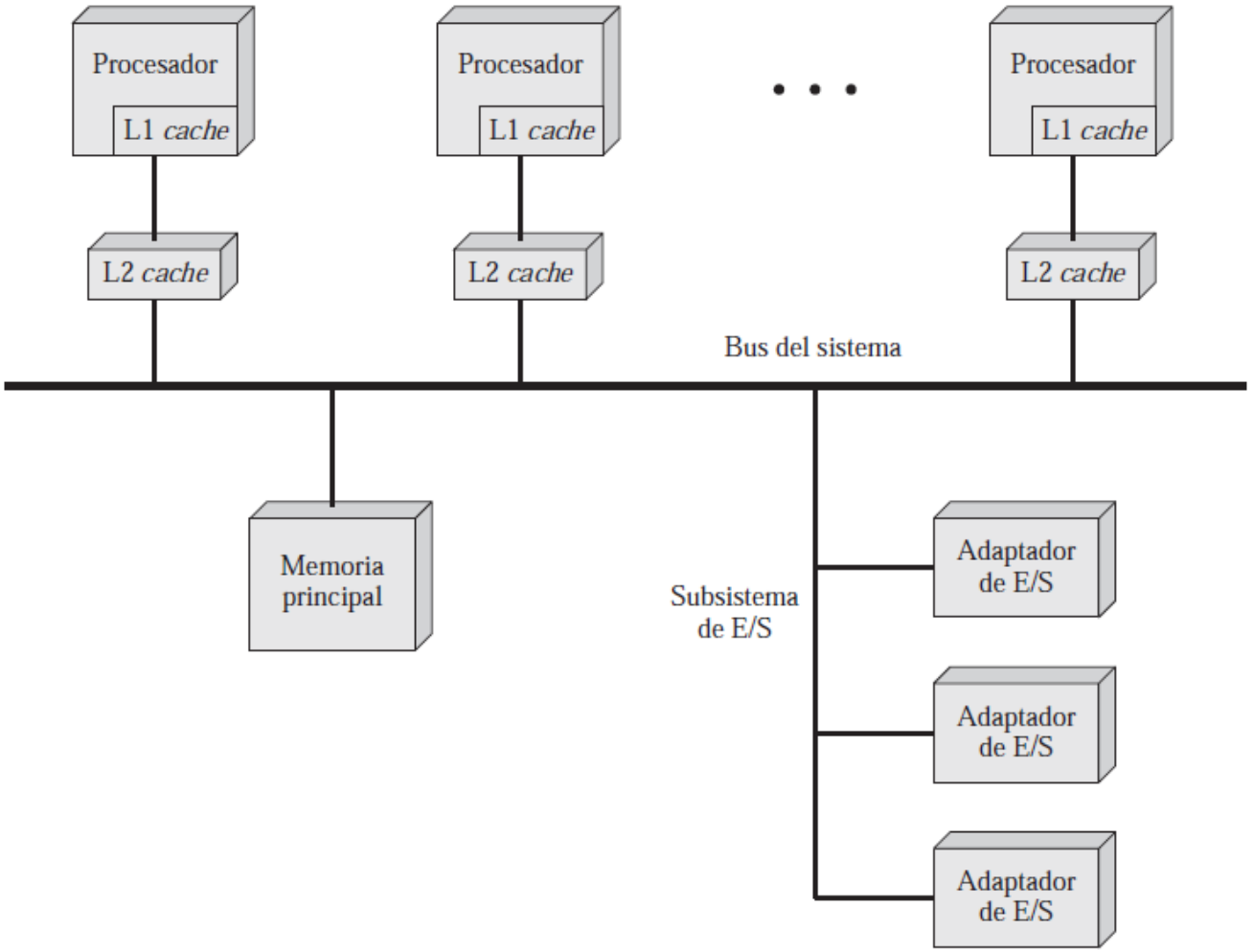
Multiprocesamiento simétrico: SMP

- **Única instrucción, único flujo de datos – *Single instruction single data* (SISD) *stream*.** Un solo procesador ejecuta una única instrucción que opera sobre datos almacenados en una sola memoria.
- **Única instrucción, múltiples flujos de datos – *Single instruction multiple data* (SIMD) *stream*.** Una única instrucción de máquina controla la ejecución simultánea de un número de elementos de proceso. Cada elemento de proceso tiene una memoria de datos asociada, de forma que cada instrucción se ejecuta en un conjunto de datos diferente a través de los diferentes procesadores. Los procesadores vectoriales y matriciales entran dentro de esta categoría.
- **Múltiples instrucciones, único flujo de datos – *Multiple instruction single data* (MISD) *stream*.** Se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferente. Esta estructura nunca se ha implementado.
- **Múltiples instrucciones, múltiples flujos de datos – *Multiple instruction multiple data* (MIMD) *stream*.** Un conjunto de procesadores ejecuta simultáneamente diferentes secuencias de instrucciones en diferentes conjuntos de datos.

Multiprocesamiento simétrico: SMP

- MIMD permite establecer un propósito general para los procesadores, porque deben ser capaces de procesar todas las instrucciones necesarias para realizar las transformaciones de datos apropiadas.
- MIMD se puede subdividir por la forma en que se comunican los procesadores. Si cada procesador tiene una memoria dedicada, cada elemento de proceso es en sí un ordenador. La comunicación entre los computadores se puede realizar a través de rutas prefijadas o bien a través de redes.
- Este sistema es conocido como un ***cluster***, o multicomputador. Si los procesadores comparten una memoria común, entonces cada procesador accede a los programas y datos almacenados en la memoria compartida, y los procesadores se comunican entre sí a través de dicha memoria; este sistema se conoce como **multiprocesador de memoria compartida**.
- El diseño de SMP y clústers es complejo, e involucra temas relativos a la organización física, estructuras de interconexión, comunicación entre procesadores, diseño del sistema operativo y técnicas de aplicaciones software.

Organización de un SMP



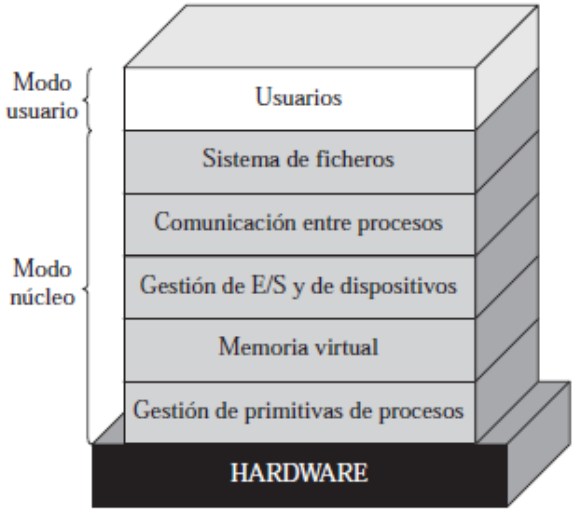
Consideraciones de diseño del SO

- **Procesos o hilos simultáneos concurrentes.** Las rutinas del núcleo necesitan ser reentrantes para permitir que varios procesadores ejecuten el mismo código del núcleo simultáneamente.
- **Planificación.** La planificación se puede realizar por cualquier procesador, por lo que se deben evitar los conflictos.
- **Sincronización**. Con múltiples procesos activos, que pueden acceder a espacios de direcciones compartidas o recursos compartidos de E/S, se debe tener cuidado en proporcionar una sincronización eficaz.
- **Gestión de memoria**. La gestión de memoria en un multiprocesador debe tratar con todos los aspectos encontrados en las máquinas uniprocesador.
- **Fiabilidad y tolerancia a fallos.** El sistema operativo no se debe degradar en caso de fallo de un procesador.

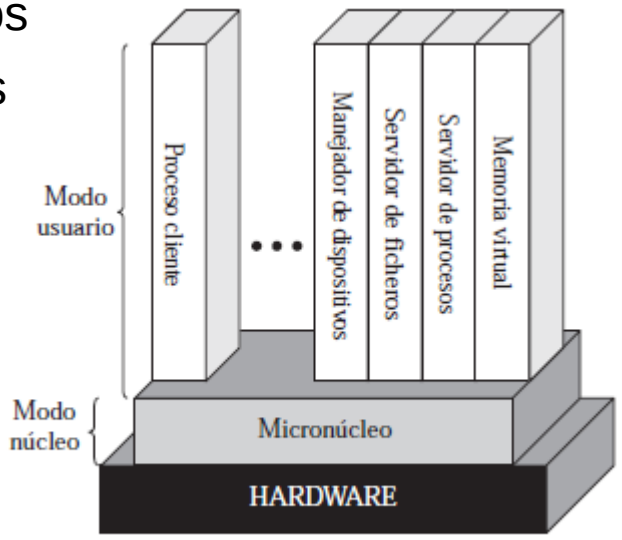
Micronúcleos

Ventajas

- Interfaces uniformes.
- Extensibilidad
- Flexibilidad
- Portabilidad
- Fiabilidad
- Soporte de sistemas distribuidos
- Soporte de sistemas operativos orientados a objetos (OOOS).



(a) Núcleo por capas



(b) Micro núcleo

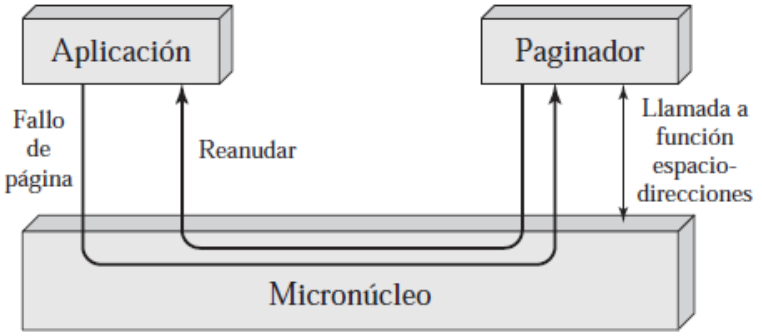
Micronúcleos

- El micronúcleo nos lleva por sí mismo al soporte de sistemas distribuidos, incluyendo clústeres controlados por sistemas operativos distribuidos.
- Cuando se envía un mensaje desde un cliente hasta un proceso servidor, el mensaje debe incluir un identificador del servicio pedido.
- Si se configura un sistema distribuido (por ejemplo, un clúster) de tal forma que todos los procesos y servicios tengan identificadores únicos, entonces habrá una sola imagen del sistema a nivel de micronúcleo.
- Un proceso puede enviar un mensaje sin saber en qué máquina reside el servicio pedido.
- Una arquitectura micronúcleo funciona bien en el contexto de un sistema operativo orientado a objetos.
- Un enfoque orientado a objetos puede servir para diseñar el micronúcleo y para desarrollar extensiones modulares para el sistema operativo.

Micronúcleos: otros aspectos

- Una potencial desventaja que se cita a menudo de los micronúcleos es la del rendimiento. Lleva más tiempo construir y enviar un mensaje a través del micronúcleo.
- **Gestión de memoria a bajo nivel**. El micronúcleo tiene que controlar el concepto hardware de espacio de direcciones para hacer posible la implementación de protección a nivel de proceso.

- **Conceder (Grant).** El propietario de un espacio de direcciones (un proceso) puede conceder alguna de sus páginas a otro proceso. El núcleo borra estas páginas del espacio de memoria del otorgante y se las asigna al proceso especificado.
- **Proyectar (Map)**. Un proceso puede proyectar cualquiera de sus páginas en el espacio de direcciones de otro proceso, de forma que ambos procesos tienen acceso a las páginas. Esto genera memoria compartida entre dos procesos. El núcleo mantiene la asignación de estas páginas al propietario inicial, pero proporciona una asociación que permite el acceso de otros procesos.
- **Limpiar (Flush)**. Un proceso puede reclamar cualquier página que fue concedida o asociada a otro proceso.



Micronúcleos: otros aspectos

- **Comunicación entre procesos (Interprocess Communication).** La forma básica de comunicación entre dos procesos o hilos en un sistema operativo con micronúcleo son los **mensajes**.
- **Gestión de E/S e interrupciones**. Con una arquitectura micronúcleo es posible manejar las interrupciones hardware como mensajes e incluir los puertos de E/S en los espacios de direcciones.
  - El micronúcleo puede reconocer las interrupciones pero no las puede manejar.
  - Este genera un mensaje para el proceso a nivel de usuario que está actualmente asociado con esa interrupción.

Bibliografía

- **CARRETERO**, Jesús, GARCÍA, Félix, DE MIGUEL, Pedro, PÉREZ, Fernando. Sistemas Operativos: una visión aplicada. McGraw-Hill, 2001.
- **STALLINGS, William. Sistemas operativos: aspectos internos y principios de diseño. 5ª Edición. Editorial Pearson Educación. 2005. ISBN: 978-84-205-4462-5.**
- **TANENBAUM**, Andrew S. Sistemas operativos modernos. 3ª Edición. Editorial Prentice Hall. 2009. ISBN: 978-607- 442-046-3.

marlon.cardenas@ufv.es

Universidad
Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior