Capítulo 3 – Hilos, SMP, Micronúcleo

Objetivos:

- Explicar el proceso de manejo del sistema operativo de la multiprogramación a través de la creación de hilos.
- Discutir el concepto de multiprocesamiento simétrico en los sistemas operativos.

Temas:

- Monohilo
- Multihilo
- SMP
- Micronúcleo

Monohilo

Un único hilo de ejecución por proceso.
Monohilo >1H=IP.

Casos:

- Un proceso con un solo hilo. Ej. MS-DOS.
- Múltiples procesos cada uno con uno solo hilo. Ej.: Unix antiguos.

Multihilo

El OS puede soportar múltiples hilos de ejecución dentro de un sólo proceso.

Casos:

- Un proceso con múltiples hilos: Java Virtual Machine.
- Múltiples procesos con múltiples hilos: Windows, Solaris, Mach, OS/2.

Ejemplos del uso de hilos:

1. Trabajos en 1er y 2do plano.

H1: muestra menús; H2: lee entrada de teclado; H3: ejecuta ordenes.

2. Procesamiento asíncrono: un hilo puede guardar 1 vez por minuto en un buffer en caso de pérdida de energía.
3. Velocidad de ejecución: cada hilo en distintos procesos.
4. Estructura modular

Un proceso en un ambiente multihilo consta de:

- espacio virtual para la imagen del proceso
- acceso protegido al procesador, otros procesos, archivos, y recursos de E/S.

Cada hilo de un proceso consta de:

- un estado de ejecución
- un contexto de hilo guardado cuando no está en ejecución
- una pila de ejecución
- un almacenamiento estático para variables locales
- acceso a memoria y recursos de sus procesos compartidos con otros hilos

Ventajas del uso de múltiples hilos dentro de los procesos:

- Menos tiempo de creación, finalización y cambio de hilo vs un proceso.
- No hay necesidad de cambio de modo del OS a núcleo para realizar comunicación entre hilos del mismo proceso. Ver. Fig. 4.2
- Mejoran el rendimiento tanto en equipo monoprocesador como en multiprocesador.

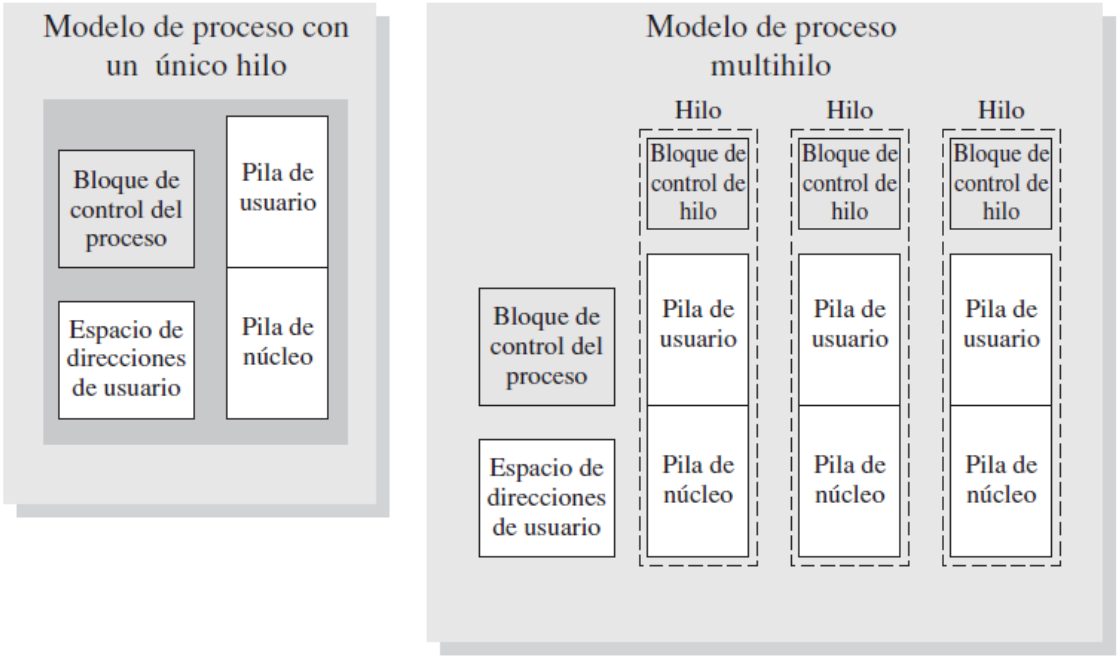


Figura 4.2. Modelos de proceso con un único hilo y multihilo.

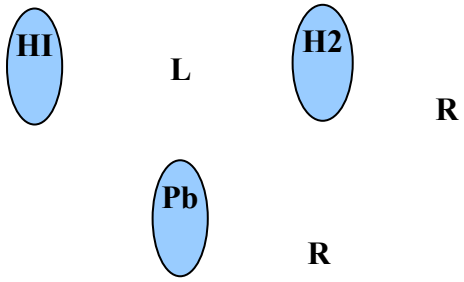
Los hilos en un monoprocesador se interconectan durante E/S o por tiempo de CPU.

Hilos a nivel de usuario – ULT	**Hilos a nivel de núcleo – KLT**
La aplicación administra los hilos; el núcleo no los reconoce, sino que lo planifica como una unidad. En la biblioteca se crea, destruye, planifica, se guarda y restaura y hay estructuras para el paso de mensajes y datos entre hilos.	El núcleo realiza toda la gestión. Caso Windows.
Ventajas:	Ventajas:

- menos carga al cambiar entre distintos hilos de un proceso porque no hay que pasar a modo núcleo; todo está en la biblioteca. - planificación puede ser específica de la aplicación dependiendo de las necesidades - pueden correr en cualquier OS. La librería de hilos es un conjunto de utilidades a nivel de aplicación compartidas por todas las aplicaciones.	- soluciona los problemas de hilos a nivel de usuario - el núcleo puede simultáneamente planificar múltiples hilos del mismo proceso en múltiples procesadores - si uno de los procesos está bloqueado, el núcleo puede planificar otro hilo del mismo proceso - las rutinas del núcleo pueden ser multihilos
Desventajas: - muchas llamadas al sistema son bloqueadoras, así que si un hilo hace una llamada al sistema bloqueará a los demás. (se resuelve con jacketing) - en una estrategia pura de hilos a nivel de usuario, sólo un hilo en un proceso puede estar en ejecución cada vez a pesar de contar con varios procesadores	Desventajas: - la transferencia de control de un hilo a otro dentro del mismo proceso requiere un cambio de modo a núcleo.

Jacketing: resuelve el bloqueo de hilos. La técnica convierte una llamada al sistema bloqueada a una no bloqueada, de manera que otros hilos puedan seguir en ejecución mientras un hilo hace una llamada al sistema.

1.

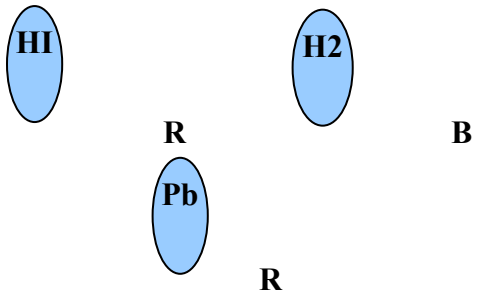


1. Se esta ejecutando el proceso B, hilo2.
2. Se esta ejecutando **H2,** hay una llamada al sistema que bloquea a **B;** el CPU pasa a otro proceso. Aunque B este Block o List, en la biblioteca H2 sigue R (aunque no este haciendo uso del CPU así lo ve la aplicación).

Obs.: Pb no esta en el CPU.

B

3. Interrupción de reloj para el proceso B, lo saca de ejecución y pasa a listo al proceso B.
4. H2 requiere de H1; H2= Block y H1=R.



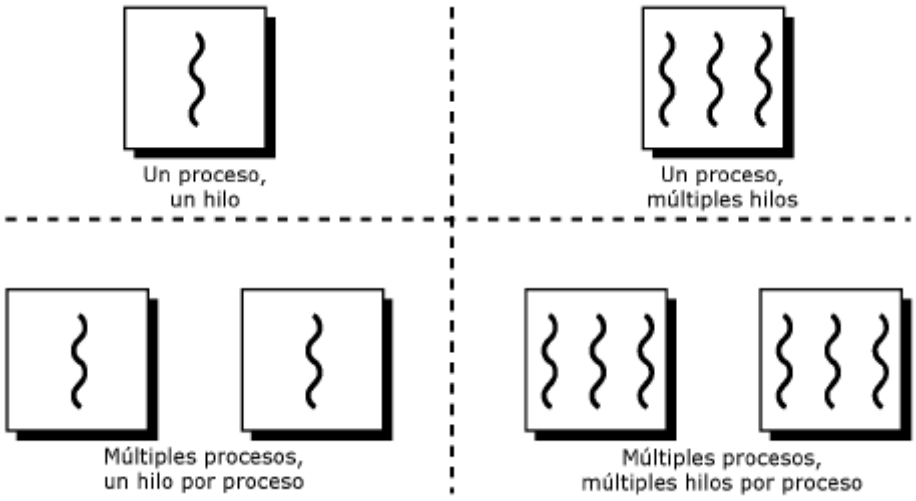
Obs.: Pb entra en ejecución así que sigue en H2 hasta que este se bloquea y se ejecuta.

3.1.2 Funcionalidad de los hilos.

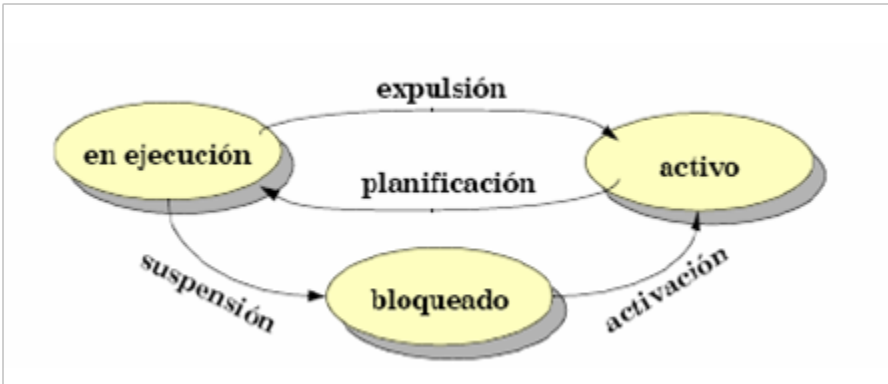
La funcionalidad de los hilos consiste en proporcionar un mecanismo eficiente para la comunicación y sincronización (comparten recursos) además evita en cierta medida los cambios de contextos.

Al igual que los procesos, los hilos poseen un estado de ejecución y pueden sincronizarse entre ellos para evitar problemas de compartimiento de recursos.

Generalmente, cada hilo tiene una tarea específica y determinada, como forma de aumentar la eficiencia del uso del procesador.



3.1.3 Estado de hilos



- **Ejecución, listo y bloqueado.**

Estados de un hilo en Unix

Un hilo puede ser en alguno de los siguientes estados:

- Listo: el hilo puede ser elegido para su ejecución.
- Standby: el hilo ha sido elegido para ser el siguiente en ejecutarse en el procesador.
- Ejecución: el hilo está siendo ejecutado.
- Espera: un hilo pasa a este estado cuando se bloquea por un suceso
- (E/S): se realiza una espera voluntaria de sincronización o alguien suspende al hilo.
- Transición: después de una espera el hilo pasa a este estado si está listo para ejecutar, pero alguno de sus recursos no está disponible aún.
- Terminado: un hilo llega a este estado cuando termina normalmente cuando su proceso padre ha terminado.

Un aspecto importante es si el bloqueo de un hilo implica el bloqueo del proceso completo. En otras palabras, si se bloquea un hilo de un proceso, ¿esto impide la ejecución de otro hilo del mismo proceso incluso si el otro hilo está en estado de Listo? Sin lugar a dudas, se pierde algo de la potencia y flexibilidad de los hilos si el hilo bloqueado bloquea al proceso entero.

Estados de un hilo en Linux

Linux no considera los hilos como tales. En Linux se crea un nuevo proceso copiando los atributos del proceso actual. Un nuevo proceso puede ser clonado para que comparta los recursos del actual, tales como archivos, gestores de la memoria virtual.

Cuando dos procesos comparten la memoria, operan en efecto como hilos dentro del mismo espacio, del mismo proceso.

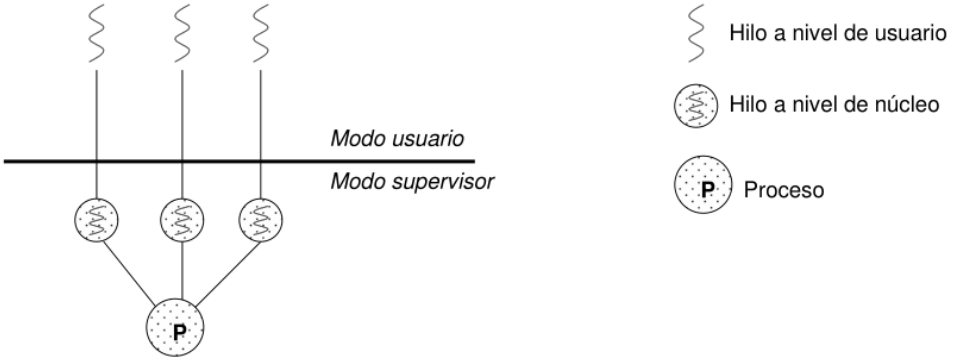
Sin embargo, no se manejan estructuras de datos para los hilos diferentes de las de los procesos, por lo que se puede argumentar que Linux no hace diferencias entre hilos y procesos.

3.1.4 Paralelismo.

Se refiere a la ejecución simultánea de varios procesos computacionales. Esto significa que se requieren varios medios de ejecución física: varios procesadores (o un procesador con varios núcleos) o varias computadoras (sistemas distribuidos) y la suficiente memoria para mantenerlos. Los procesos pueden estar relacionados entre ellos, para realizar una misma tarea, o no (Patricio, s.f.).

El paralelismo está relacionado con la capacidad del sistema en el que se ejecuta el programa, con sus recursos disponibles y que el software lo pueda aprovechar (Patricio, s.f.).

3.1.5 Hilos a nivel de usuario y de kernel



Hilos a nivel de usuario: son implementados en alguna librería. Estos hilos se gestionan sin soporte del SO, el cual solo reconoce un hilo de ejecución.

- En un entorno ULT puro, la aplicación gestiona todo el trabajo de los hilos y el núcleo no es consciente de la existencia de los mismos. La Figura 4.6 muestra el enfoque ULT. Cualquier aplicación puede programarse para ser multihilo a través del uso de una biblioteca de hilos, que es un paquete de rutinas para la gestión de ULT. La biblioteca de hilos contiene código para la creación y destrucción de hilos, para paso de mensajes y datos entre los hilos, para planificar la ejecución de los hilos, y para guardar y restaurar el contexto de los hilos.

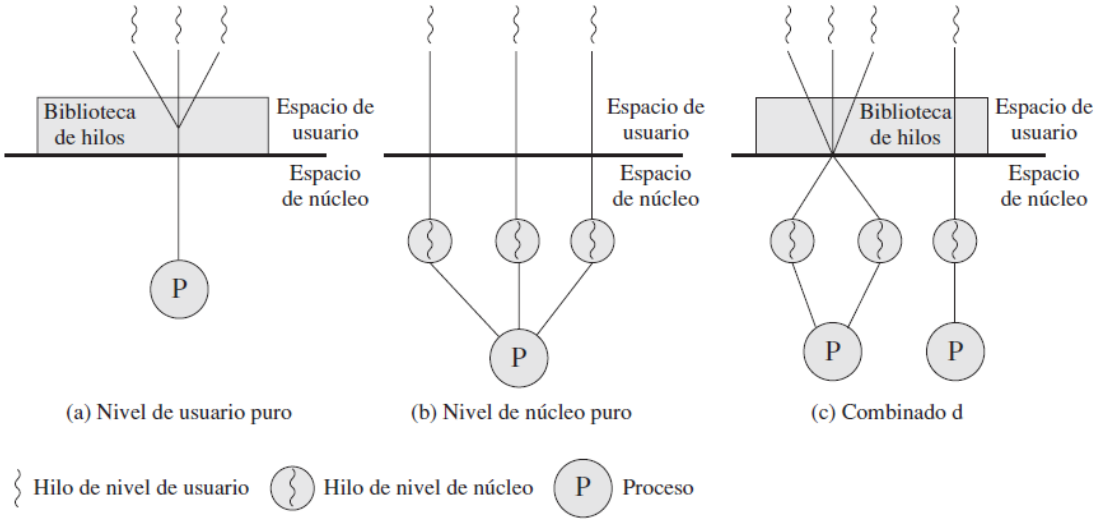


Figura 4.6. Hilos de nivel de usuario y de nivel de núcleo.

Hilos a nivel de kernel: el SO es quien crea, planifica y gestiona los hilos. Se reconocen tantos hilos como se hayan creado. Los hilos a nivel de usuario tienen como beneficio que su cambio de contexto es más sencillo que el cambio de contexto entre hilos de kernel. A demás, se pueden implementar aún si el SO no utiliza hilos a nivel de kernel. Otro de los beneficios consiste en poder planificar diferente a la estrategia del SO. Los hilos a nivel de kernel tienen como gran beneficio poder aprovechar mejor las arquitecturas multiprocesadores, y que proporcionan un mejor tiempo de respuesta, ya que, si un hilo se bloquea, los otros pueden seguir ejecutando.

- En un entorno KLT puro, el núcleo gestiona todo el trabajo de gestión de hilos. No hay código de gestión de hilos en la aplicación, solamente una interfaz de programación de aplicación (API) para acceder a las utilidades de hilos del núcleo. Windows es un ejemplo de este enfoque.

El uso de ULT en lugar de KLT, presenta las siguientes ventajas:

1. El cambio de hilo no requiere privilegios de modo núcleo porque todas las estructuras de datos de gestión de hilos están en el espacio de direcciones de usuario de un solo proceso. Por consiguiente, el proceso no cambia a modo núcleo para realizar la gestión de hilos. Esto ahorra la sobrecarga de dos cambios de modo (usuario a núcleo; núcleo a usuario).
2. La planificación puede especificarse por parte de la aplicación. Una aplicación se puede beneficiar de un simple algoritmo de planificación cíclico, mientras que otra se podría beneficiar de un algoritmo de planificación basado en prioridades. El algoritmo de planificación se puede hacer a medida sin tocar el planificador del sistema operativo.
3. Los ULT pueden ejecutar en cualquier sistema operativo. No se necesita ningún cambio en el nuevo núcleo para dar soporte a los ULT. La

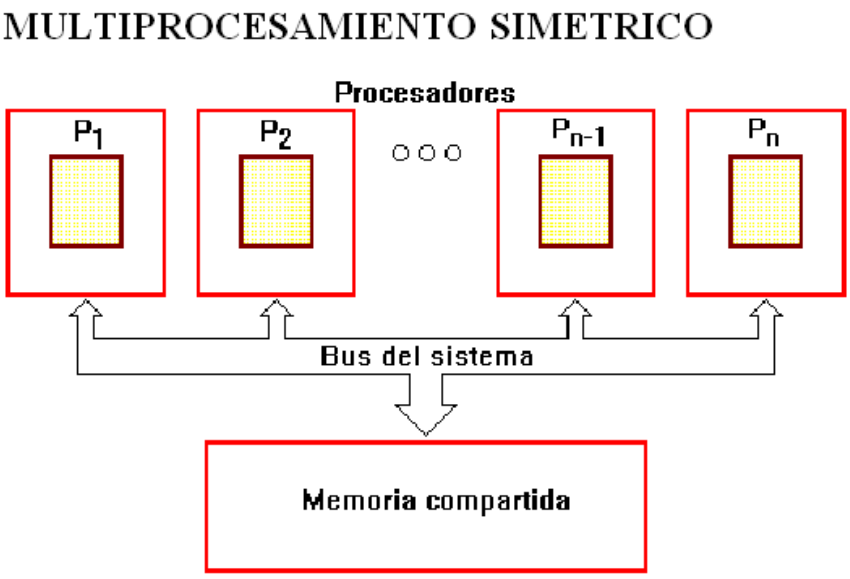
biblioteca de los hilos es un conjunto de utilidades a nivel de aplicación que comparten todas las aplicaciones.

Hay dos desventajas de los ULT en comparación con los KLT:

1. En un sistema operativo típico muchas llamadas al sistema son bloqueantes. Como resultado, cuando un ULT realiza una llamada al sistema, no sólo se bloquea ese hilo, sino que se bloquean todos los hilos del proceso.
2. En una estrategia pura ULT, una aplicación multihilo no puede sacar ventaja del multiproceso. El núcleo asigna el proceso a un solo procesador al mismo tiempo. Por consiguiente, en un determinado momento sólo puede ejecutar un hilo del proceso. En efecto, tenemos multiprogramación a nivel de aplicación con un solo proceso. Aunque esta multiprogramación puede dar lugar a una mejora significativa de la velocidad de la aplicación, hay aplicaciones que se podrían beneficiar de la habilidad de ejecutar porciones de código de forma concurrente.

Multiproceso simétrico

El multiprocesamiento simétrico se entiende como los sistemas que incluyen más de un procesador. Son capaces de ejecutar diversos procesos de forma simultánea y, además, comparten una misma memoria para el cumplimiento de sus funciones (Team, 2023).



Características del multiprocesamiento simétrico

- Dentro de las características y propiedades destacables del multiprocesamiento simétrico, se encuentra su utilidad para la adición de procesadores, memoria y otros recursos y componentes que permiten el incremento del rendimiento en los sistemas.

- De la misma forma, este tipo de multiprocesamiento se caracteriza porque cada procesador lleva a cabo la ejecución de las tareas en el sistema operativo. Así, se toman los procesos de una cola preparada en común o privada para cada uno de los procesadores.

- Como propiedad del multiprocesamiento simétrico aparece también que todo su procesador mantiene la misma arquitectura y que sus procesadores incluyen la capacidad de comunicarse con otros a través del recurso de memoria compartida.

Se refiere a la arquitectura hardware del sistema multiprocesador y al comportamiento del sistema operativo que utiliza dicha arquitectura. Un SMP es un computador con las siguientes características:
1) Tiene dos o más procesadores similares de capacidades comparables.
2) Los procesadores comparten la memoria principal y la E/S, y están interconectados mediante un bus u otro tipo de sistema de interconexión, de manera que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores.
3) Todos los procesadores comparten los dispositivos de E/S, pero pueden hacerlo bien a través de los mismos canales, o bien a través de otros caminos de acceso al mismo dispositivo.
4) Todos los procesadores pueden desempeñar las mismas funciones (de ahí el término simétrico).
5) El sistema está controlado por un sistema operativo que posibilita la interacción entre los procesadores y sus programas.

Objetivo de la especificación MP

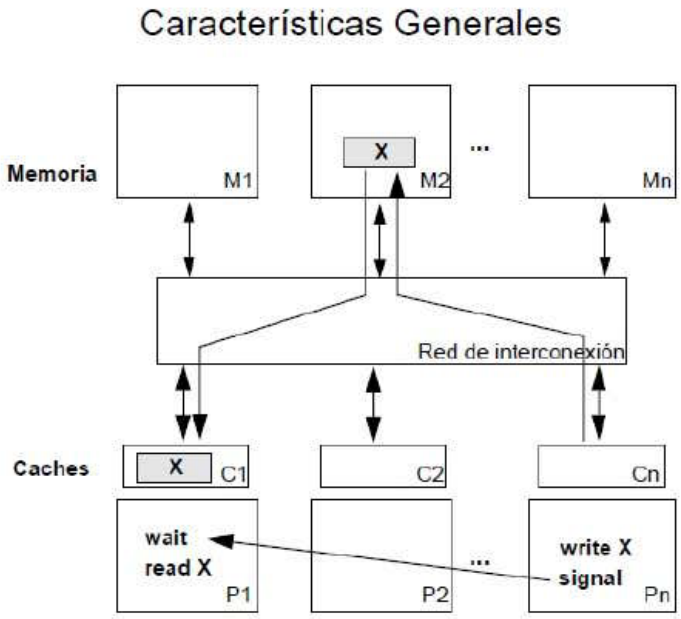
La meta más importante de esta norma es realizar una ampliación de la plataforma PC/AT mediante la cual sea posible diseñar equipos multiprocesador manteniendo una compatibilidad total con el hardware y el software existente (Galàn, 1996).

Mediante la consecución de este objetivo anteriormente mencionado, es posible continuar ejecutando todas las aplicaciones existentes para la plataforma PC/AT y, a través de la utilización de sistemas operativos y aplicaciones para multiproceso, conseguir importantes incrementos en la productividad de los equipos.

Para construir un sistema compatible con esta especificación es necesario, sin embargo, realizar modificaciones en elementos clave del ordenador, como la BIOS del equipo, que debe construir una tabla de datos que presente el hardware de una forma clara a los controladores de dispositivo o a la capa de abstracción de hardware del sistema operativo (Galàn, 1996).

3.2.1 Arquitectura SMP

La arquitectura SMP (Multi-procesamiento simétrico, también llamada UMA, de Uniform Memory Access), se caracteriza por el hecho de que varios microprocesadores comparten el acceso a la memoria. Todos los microprocesadores compiten en igualdad de condiciones por dicho acceso, de ahí la denominación "simétrico". Los sistemas SMP permiten que cualquier procesador trabaje en cualquier tarea sin importar su localización en memoria; con un propicio soporte del sistema operativo, estos sistemas pueden mover fácilmente tareas entre los procesadores para garantizar eficientemente el trabajo.



Una computadora SMP se compone de microprocesadores independientes que se comunican con la memoria a través de un bus compartido. Dicho bus es un recurso de uso común. Por tanto, debe ser arbitrado para que solamente un microprocesador lo use en cada instante de tiempo. Si las computadoras con un solo microprocesador tienden a gastar considerable tiempo esperando a que lleguen los datos desde la memoria, SMP empeora esta situación, ya que hay varios parados en espera de datos (Gustavo, s.f.).

La forma más común de categorizar estos sistemas es la taxonomía de sistemas de procesamiento paralelo introducida por Flynn [FLYN72]. Flynn propone las siguientes categorías de sistemas de computadores:

- **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.

En un **multiprocesador simétrico (Symmetric Multiprocessor, SMP)**, el núcleo puede ejecutar en cualquier procesador, y normalmente cada procesador realiza su propia planificación del conjunto disponible de procesos e hilos. El núcleo puede construirse como múltiples procesos o múltiples hilos, permitiéndose la ejecución de partes del núcleo en paralelo. El enfoque SMP complica al sistema operativo, ya que debe asegurar que dos procesadores no seleccionan un mismo proceso y que no se pierde ningún proceso de la cola. Se deben emplear técnicas para resolver y sincronizar el uso de los recursos.

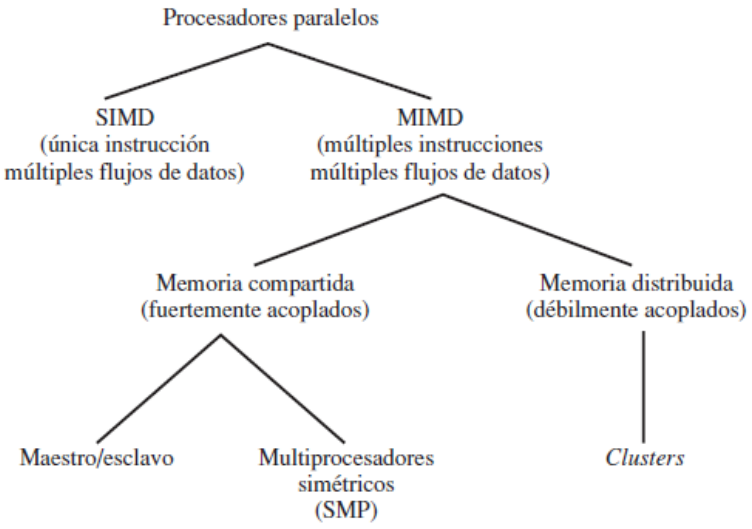


Figura 4.8. Arquitectura de procesadores paralelos.

El diseño de SMP y clusters 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.

3.2.2 Organización SMP

La Figura 4.9 muestra la organización general de un SMP. Existen múltiples procesadores, cada uno de los cuales contiene su propia unidad de control, unidad aritmético-lógica y registros. Cada procesador tiene acceso a una memoria principal compartida y dispositivos de E/S a través de algún mecanismo de interconexión; el bus compartido es común a todos los procesadores.

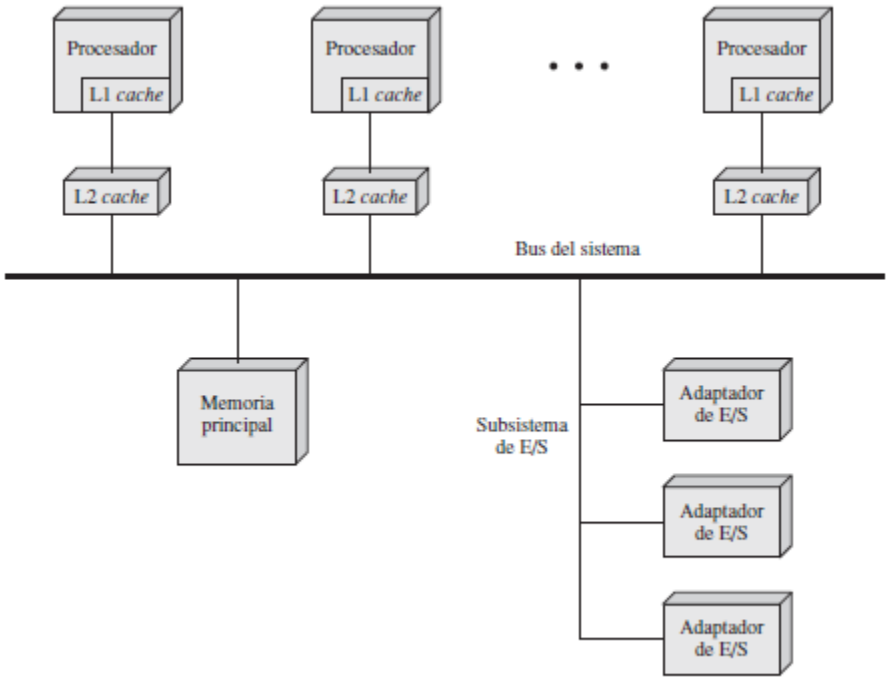
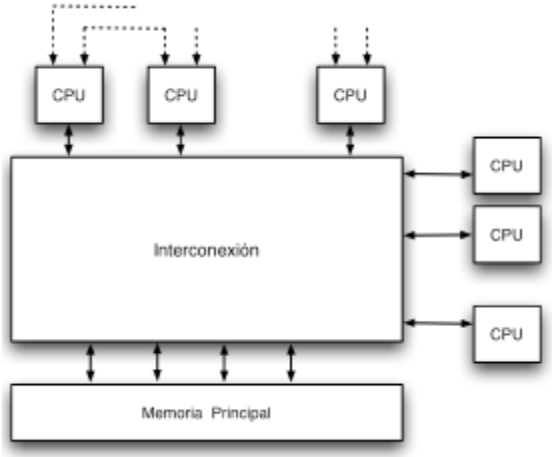


Figura 4.9. Organización de los multiprocesadores simétricos.

Los procesadores se pueden comunicar entre sí a través de la memoria (mensajes e información de estado dejados en espacios de memoria compartidos). Los procesadores han de poder intercambiarse señales directamente. A menudo la memoria está organizada de tal manera que se pueden realizar múltiples accesos simultáneos a bloques separados.



Existen dos o más CPUs, cada una de las cuales contiene:
a) unidad de control
b) ALU
c) registros
d) posiblemente caché.
Cada CPU tiene acceso a una memoria principal compartida y a los dispositivos I/O a través de un mecanismo de interconexión. Los procesadores pueden comunicarse entre ellos a través de la memoria (mensajes e información de estados almacenados en áreas comunes).

La organización de un sistema multiprocesador puede clasificarse de la siguiente forma:
• Tiempo compartido o Bus Común.
• Memoria Multipuerto.
• Unidad de Control Central.

Bus de tiempo compartido:

- Direccionamiento: Se pueden distinguir los módulos del bus para determinar la fuente y el destino de los datos.
- Arbitraje: Existe un mecanismo para arbitrar peticiones de control del bus, utilizando algún tipo de esquema de prioridades. Los módulos I/O también pueden funcionar temporalmente como master.
- Tiempo Compartido: Cuando un módulo está controlando el bus, los módulos restantes no están autorizados y deben suspender, si es necesario, la operación hasta que se les asigne el acceso al bus.

Las principales ventajas de esta estructura son:

• Simplicidad, ya que la estructura es la misma que en un sistema uniprocesador.
• Flexibilidad: Es fácil expandir el sistema añadiendo más CPUs.
• Fiabilidad: El bus es esencialmente un medio pasivo, por lo que en principio no debe producir fallos en el sistema.

Memoria multipuerto:

La aproximación de memoria multipuerto permite el acceso independiente y directo a los módulos de memoria principal por parte de cada CPU y módulo I/O. Se necesita una lógica asociada a la memoria para resolver conflictos de acceso. El método más utilizado es asignar permanentemente prioridades designadas a cada puerto de memoria.

Unidad Central Control:
La unidad central de control proporciona canales de datos separados para cada sentido entre módulos independientes: CPU, memoria y I/O. El controlador memoriza las peticiones e implementa funciones de arbitraje y temporización. Puede pasar también mensajes de control y estado entre CPUs, y alertar de modificación de cachés.

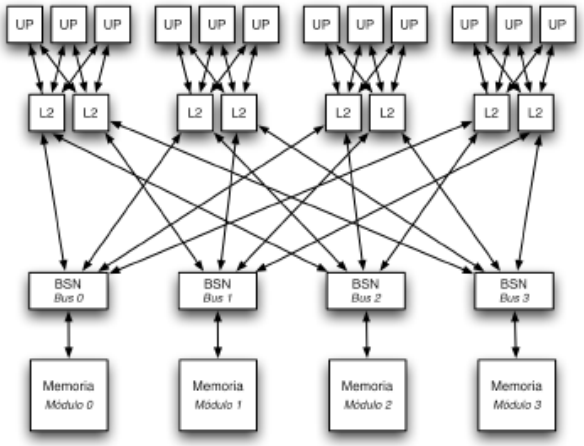
3.2.3 Consideraciones

Consideraciones de diseño de sistemas operativos multiprocesador

Las principales claves de diseño incluyen las siguientes características:

- **Proceso o hilo 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. Si se utiliza multihilo a nivel de núcleo, existe la posibilidad de planificar múltiples hilos del mismo proceso simultáneamente en múltiples procesadores.
- **Sincronización:** La sincronización es un servicio que fuerza la exclusión mutua y el orden de los eventos.
- **Gestión de memoria:** Además, el sistema operativo necesita explotar el paralelismo hardware existente, como las memorias multipuerto, para lograr el mejor rendimiento. Los mecanismos de paginación de los diferentes procesadores deben estar coordinados para asegurar la consistencia cuando varios procesadores comparten una página o segmento y para decidir sobre el reemplazo de una página.
- **Fiabilidad y tolerancia a fallos.** El sistema operativo no se debe degradar en caso de fallo de un procesador. El planificador y otras partes del sistema operativo deben darse cuenta de la pérdida de un procesador y reestructurar las tablas de gestión apropiadamente.

Grandes Computadores SMP (IBM S/390)
La mayoría de los PC y estaciones de trabajo de tipo SMP utilizan una estrategia de interconexión basada en bus. Sin embargo, existen otro tipo de organizaciones donde el mecanismo de interconexión no es un bus. Por ello, resulta ilustrativo analizar una aproximación alternativa al bus común, que se utiliza en las implementaciones más recientes de la familia de grandes computadores (mainframes) IBM S/390.



Muestra un esquema que corresponde a la organización general del SMP S/390. Esta familia de sistemas es escalable, e incluye, desde computadores monoprocesador con un módulo de memoria principal, en su configuración más básica, hasta sistemas con diez procesadores y cuatro módulos de memoria, en la gama alta.

Organización micronúcleo

Implementan en su núcleo únicamente la planificación de procesos, la gestión de interrupciones (la parte básica fundamental de la gestión de E/S que necesariamente se tiene que realizar en modo privilegiado) y la comunicación entre procesos. Por tanto, la administración de memoria principal, la gestión de la E/S y la gestión de ficheros se realiza en modo usuario. En este tipo de sistema operativo hay procesos especiales propios del sistema operativo que implementan dichas funcionalidades en modo usuario y se denominan proceso servidor.

Se caracterizan por disponer de un núcleo que implementa únicamente:

- Planificación de procesos
- Mecanismo de comunicación entre procesos
- Gestión de interrupciones

Además, existen procesos servidores que se ejecutan en modo no privilegiado del procesador - que, por supuesto, se ejecutan fuera del espacio del núcleo del sistema operativo - y que implementan los siguientes componentes:

- Administración de memoria principal
- Administración de ficheros
- Gestión de dispositivos de entrada/salida.

Siguiendo este esquema, cuando un proceso cualquiera solicita un servicio a través de una llamada al sistema, el micronúcleo canaliza la petición al proceso servidor correspondiente. Dicha comunicación se realiza mediante mensajería.
Los procesos no tienen que distinguir entre nivel de núcleo y usuario porque los servicios son provistos por el paso de mensajes.

La desventaja es que toma más tiempo en construir y enviar un mensaje por el micronúcleo y aceptar y decodificar la respuesta que hacer un simple servicio de llamada.

La filosofía existente en el micronúcleo es que solamente las funciones absolutamente esenciales del sistema operativo estén en el núcleo. Los servicios y aplicaciones menos esenciales se construyen sobre el micronúcleo y se ejecutan en modo usuario. Aunque la filosofía de qué hay dentro y qué hay fuera del micronúcleo varía de un diseño a otro, la característica general es que muchos servicios que tradicionalmente habían formado parte del sistema operativo ahora son subsistemas externos que interactúan con el núcleo y entre ellos mismos; algunos ejemplos son: manejadores de dispositivos, servidores de archivos, gestores de memoria virtual, sistemas de ventana y servicios de seguridad.

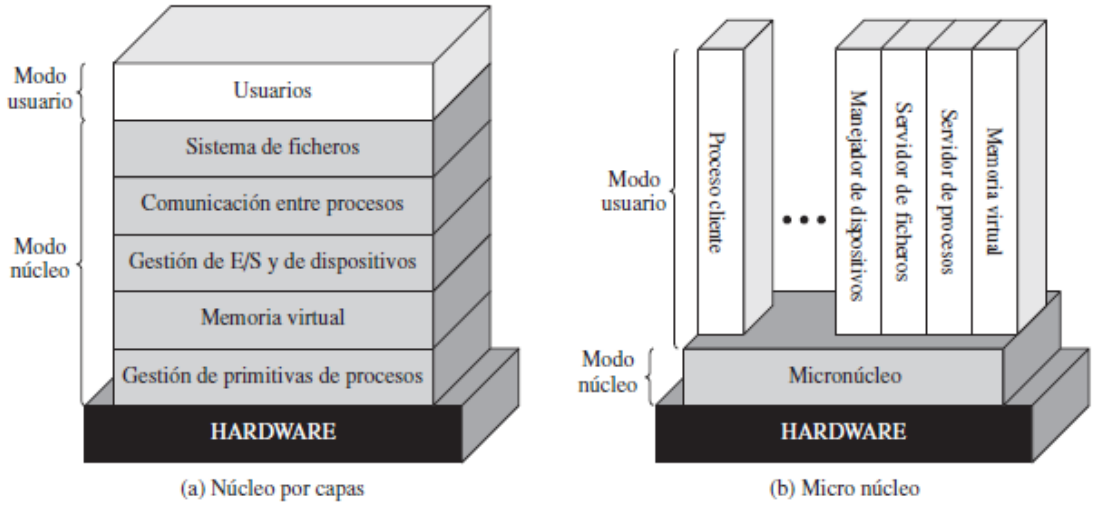


Figura 4.10. Arquitectura del núcleo.

La arquitectura del micronúcleo reemplaza la tradicional estructura vertical y estratificada en capas por una horizontal (Figura 4.10). Los componentes del sistema operativo externos al micronúcleo se implementan como servidores de procesos; interactúan entre ellos dos a dos, normalmente por paso de mensajes a través del micronúcleo. De esta forma, el micronúcleo funciona como un intercambiador de mensajes: válida mensajes, los pasa entre los componentes, y concede el acceso al hardware. El micronúcleo también realiza una función de protección; previene el paso de mensajes a no ser que el intercambio esté permitido.

Beneficios de una organización micronúcleo:

- Interfaces uniformes
- Extensibilidad
- Flexibilidad
- Portabilidad
- Fiabilidad

- Soporte de sistemas distribuidos
- Soporte de sistemas operativos orientados a objetos (OOOS)

Rendimiento del micronúcleo

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, y aceptar y decodificar la respuesta, que hacer una simple llamada a un servicio. Sin embargo, también son importantes otros factores, de forma que es difícil generalizar sobre la desventaja del rendimiento, si es que la hay.

Diseño del micronúcleo

El micronúcleo debe incluir aquellas funciones que dependen directamente del hardware y aquellas funciones necesarias para mantener a los servidores y aplicaciones operando en modo usuario. Estas funciones entran dentro de las categorías generales de gestión de memoria a bajo nivel, intercomunicación de procesos (IPC), y E/S y manejo de interrupciones.

- **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. Con tal de que el micronúcleo se responsabilice de la asignación de cada página virtual a un marco físico, la parte principal de gestión de memoria, incluyendo la protección del espacio de memoria entre procesos, el algoritmo de reemplazo de páginas y otra lógica de paginación, pueden implementarse fuera del núcleo.

  [LIED95] recomienda un conjunto de tres operaciones de micronúcleo que pueden dar soporte a la paginación externa y a la gestión de memoria virtual:
  • **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.

- **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. Más bien, genera un mensaje para el proceso a nivel de usuario que está actualmente asociado con esa interrupción. De esta forma, cuando se habilita una interrupción, se asigna un proceso de nivel de usuario a esa interrupción y el núcleo mantiene las asociaciones. La transformación de las interrupciones en mensajes las debe realizar el micronúcleo, pero el micronúcleo no está relacionado con el manejo de interrupciones específico de los dispositivos.

3.4 Hilos y Multiproceso Simétrico (SMP) en Linux.

Los hilos o threads son rutinas de código que corren de manera concurrente dentro de un mismo proceso.
Linux no considera los hilos como tales, se refiere a ellos como tareas más que como hilos. En Linux se crea un nuevo proceso copiando los atributos del proceso actual. Un nuevo proceso puede ser clonado para que comparta los recursos del actual, tales como archivos, gestores de señales o la memoria virtual.

Un proceso, o tarea, en Linux se representa por una estructura de datos task_struct, que contiene información de diversas categorías:

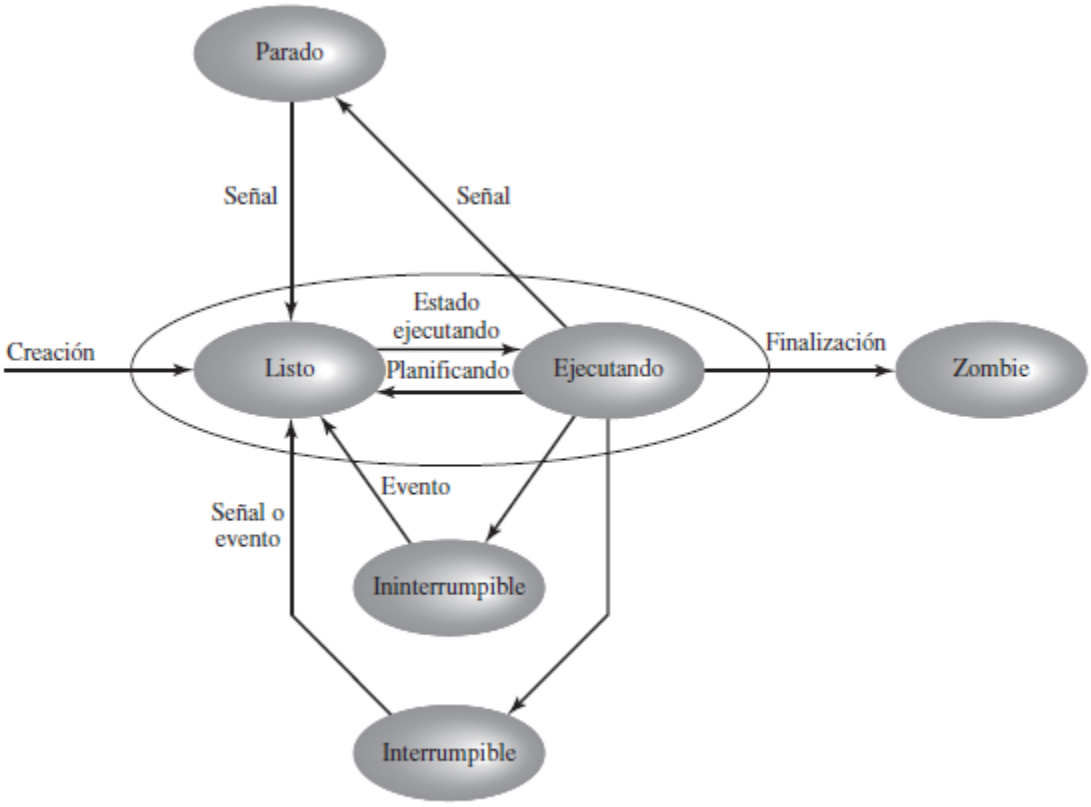


Figura 4.18. Modelo de procesos e hilos en Linux.

3.5 Hilos y SMP en UNIX.

Solaris hace uso de cuatros conceptos independientes relativos a hilos:

Proceso: este es el proceso UNIX convencional e incluye el espacio de direcciones de usuarios, la pila, y el bloque de control de procesos.

Hilos a nivel de usuario: implementados en el espacio de direcciones de un proceso por medio de una biblioteca de hilos, esos son invisibles para el sistema operativo. Los hilos a nivel de usuario (ULT) son la interfaz para el paralelismo de aplicaciones.

Procesos ligeros: un proceso ligero (LWP) puede verse como una correspondencia entre ULT e hilos del núcleo. Cada LWP soporta uno o más ULT y los hace corresponder con un hilo del núcleo. El núcleo planifica los LWP independientemente y puede ejecutar en paralelo sobre multiprocesadores.
Hilos del núcleo: son las entidades básicas de planificación y expedición en cada uno de los procesadores del sistema. Un LWP es visible para la aplicación dentro del proceso. De este modo, las estructuras de datos LWP existen dentro de los respectivos espacios de direcciones de los procesos. Al mismo tiempo, cada LWP este confinado a un único hilo del núcleo y la estructura de datos de este hilo del núcleo se mantiene dentro del espacio de direcciones del núcleo.

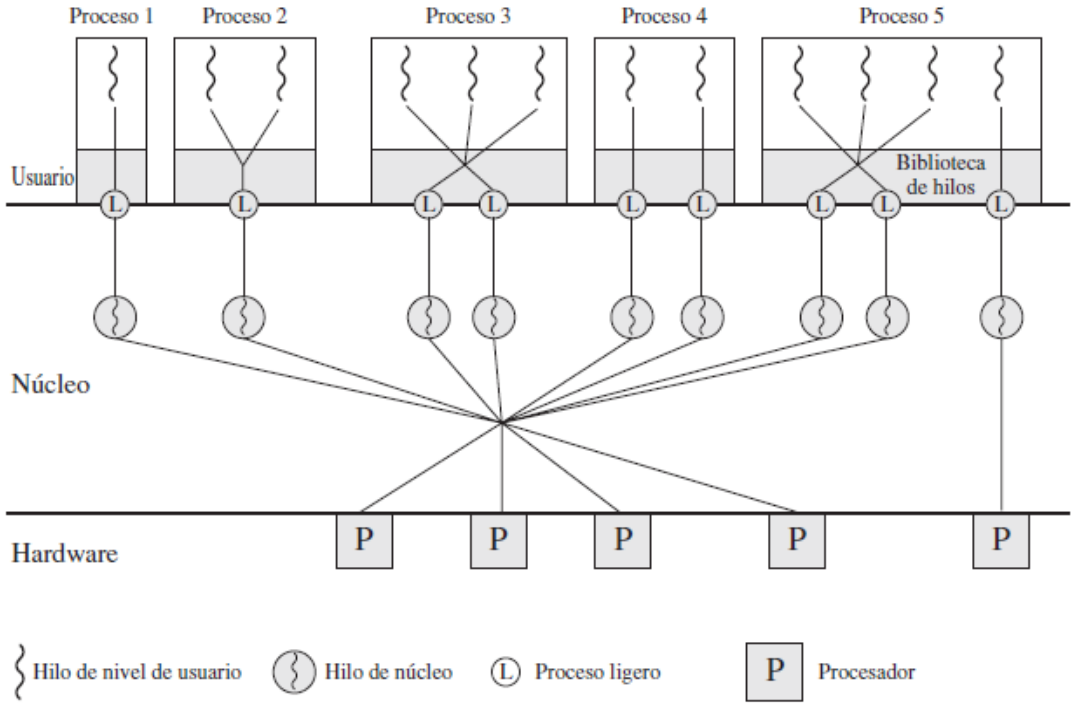


Figura 4.15. Ejemplo de arquitectura multihilo de Solaris.

La Figura 4.15 muestra la relación entre estas cuatro entidades. Nótese que hay siempre un hilo de núcleo por cada LWP. Un LWP es visible dentro de un proceso de la aplicación. De esta forma, las estructuras de datos LWP existen dentro del espacio de

direcciones del proceso respectivo. Al mismo tiempo, cada LWP está vinculado a un único hilo de núcleo activable, y la estructura de datos para ese hilo de núcleo se mantiene dentro del espacio de direcciones del núcleo.

Algunas referencias adicionales:
http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/capitulo7.html
http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/capitulo5.html
http://www.boost.org/doc/html/threads.html
http://www.flipcode.com/articles/article_multithreading.shtml

Lectura adicional (aplicativo):
Hilos en Windows
http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art130.asp

Referencias

(s.f.). Obtenido de https://www.monografias.com/trabajos26/estados-proceso-hilos/estados-proceso-hilos

Galàn, S. (1 de enero de 1996). *Multiproceso simétrico*. Obtenido de https://www.dealerworld.es/archive/multiproceso-simetrico

Gustavo, L. C. (s.f.). *La arquitectura SMP*. Obtenido de https://es.scribd.com/document/106857115/La-Arquitectura-SMP#

Patricio, H. (s.f.). *La diferencia entre concurrencia y paralelismo*. Obtenido de https://blog.thedojo.mx/2019/04/17/la-diferencia-entre-concurrencia-y-paralelismo.html#:~:text=Se%20refiere%20a%20la%20ejecuci%C3%B3n,la%20suficiente%20memoria%20para%20mantenerlos.

Team, k. (1 de marzo de 2023). *¿Qué es el multiprocesamiento simétrico?* Obtenido de https://keepcoding.io/blog/que-es-multiprocesamiento-simetrico/#:~:text=El%20multiprocesamiento%20sim%C3%A9trico%20se%20entiende,el%20cumplimiento%20de%20sus%20funciones.

Vasquez, C. (13 de julio de 2016). *Funcionalidad de los hilos*. Obtenido de https://sistemasoperativos05blog.wordpress.com/2016/07/13/funcionalidad-de-los-hilos/