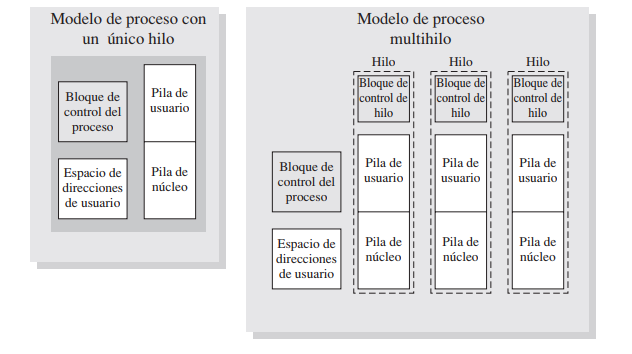
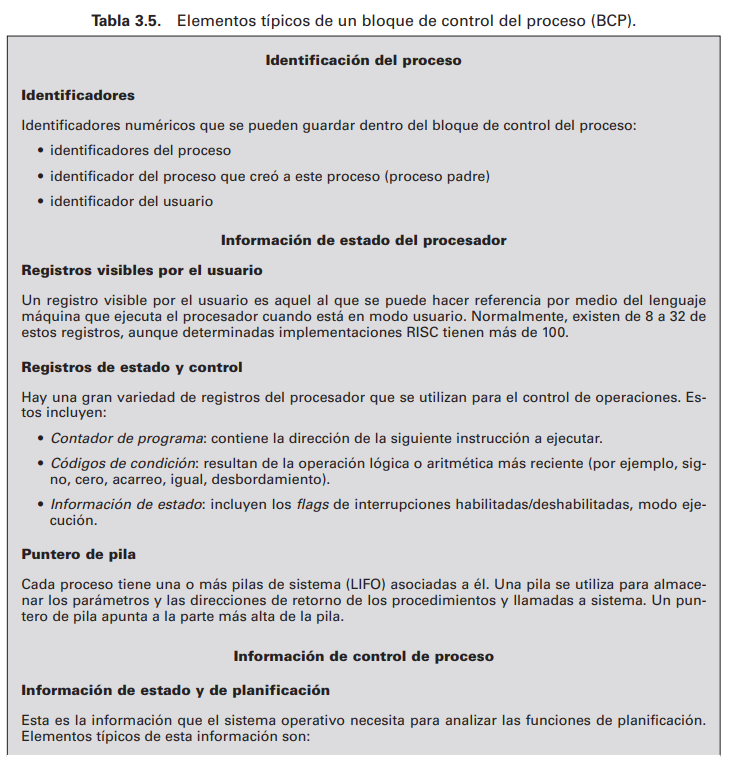
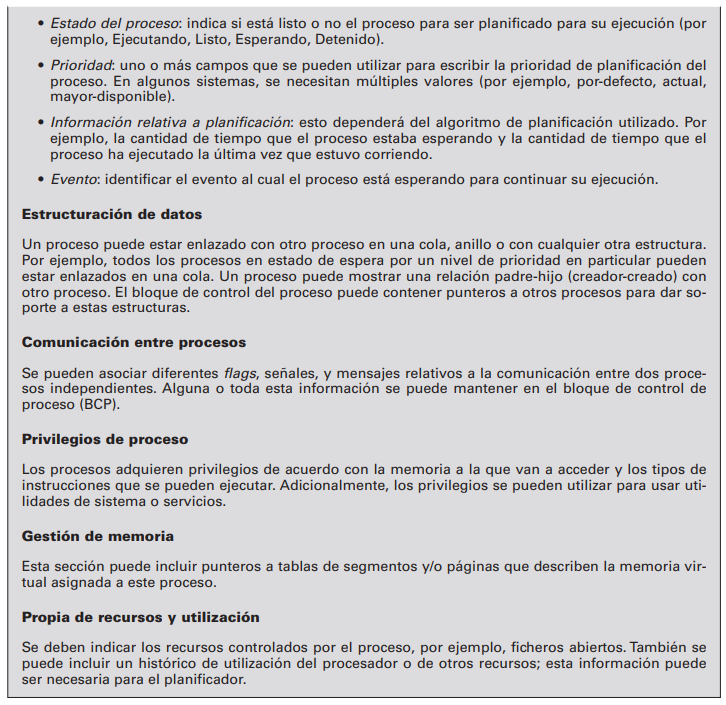
**4.1. La Tabla 3.5 enumera los elementos típicos que se encuentran en un bloque de control de proceso para un sistema operativo monohilo. De éstos, ¿cuáles deben pertenecer a un bloque de control de hilo y cuáles deben pertenecer a un bloque de control de proceso para un sistema multihilo? ~~🡪 Chequear~~** **Respuesta:**

****





**Elementos que pueden pertenecer al BCP de un proceso multihilo:**

1) Identificación del proceso: Los identificadores del proceso, el identificador del proceso padre y el identificador del usuario generalmente se mantienen a nivel de proceso, ya que son atributos del proceso en su conjunto y no de hilos individuales.

2) Información de estado del procesador: Los registros visibles por el usuario y los registros de estado y control generalmente se mantienen a nivel de hilo, ya que cada hilo tiene su propio estado de ejecución y sus registros de procesador.

3) Información de control de proceso:

- Estado del proceso: El estado del proceso podría incluir el estado de cada hilo dentro del proceso, indicando si están listos, esperando, en ejecución, etc.

- Prioridad: La prioridad podría aplicarse a cada hilo dentro del proceso, permitiendo una planificación independiente para cada hilo.

- Información relativa a la planificación: Esta información puede ser relevante para cada hilo individualmente, como el tiempo de espera y el tiempo de ejecución.

- Evento: Los eventos por los que los hilos están esperando podrían mantenerse a nivel de proceso o de hilo, dependiendo de cómo se diseñe el sistema.

4) Estructuración de datos: Los punteros a otros procesos o hilos podrían mantenerse a nivel de proceso si se refieren a relaciones entre procesos o a nivel de hilo si se refieren a la relación padre-hijo entre hilos.

6) Privilegios de proceso: Los privilegios de acceso a memoria y tipos de instrucciones pueden aplicarse a nivel de proceso, ya que generalmente se relacionan con el proceso en su conjunto.

7) Gestión de memoria: Los punteros a tablas de segmentos y/o páginas podrían mantenerse a nivel de proceso, ya que la memoria virtual asignada generalmente se comparte entre todos los hilos del proceso.

8) Propia de recursos y utilización: La indicación de recursos controlados por el proceso, como los archivos abiertos, y el historial de utilización del procesador o de otros recursos, generalmente se mantiene a nivel de proceso.

**Elementos que deben pertenecer al bloque de control del hilo (BCH) para un sistema multihilo:**

2) Información de estado del procesador: Los registros visibles por el usuario y los registros de estado y control generalmente se mantienen a nivel de hilo, ya que cada hilo tiene su propio estado de ejecución y sus registros de procesador.

3) Información de control de proceso:

* + Estado del proceso: Aunque el estado del proceso se puede mantener a nivel de proceso, se necesita una versión específica para cada hilo dentro del proceso.
  + Prioridad: La prioridad de planificación de cada hilo debe ser mantenida individualmente.
  + Información relativa a la planificación: Esta información debe mantenerse para cada hilo individualmente.
  + Evento: Los eventos por los que cada hilo está esperando deben mantenerse a nivel de hilo.

**4.2. Enumere las razones por las que un cambio de contexto entre hilos puede ser más barato que un cambio de contexto entre procesos. ~~🡪 Chequear~~** Chequeado

**1. Memoria compartida:** Los hilos dentro de un proceso comparten el mismo espacio de memoria, lo que significa que no es necesario copiar datos entre ellos durante un cambio de contexto. En cambio, los procesos tienen su propio espacio de memoria y, por lo tanto, necesitan copiar datos si desean compartir información.

**2. Menor sobrecarga del sistema operativo:** Cambiar entre hilos generalmente implica cambiar de contexto dentro del mismo espacio de direcciones de memoria, lo que puede ser gestionado más eficientemente por el sistema operativo. Por otro lado, cambiar entre procesos implica cambiar entre espacios de direcciones diferentes, lo que generalmente implica una sobrecarga adicional, como el cambio de tablas de páginas y la limpieza de la caché de memoria.

**3. Comunicación más eficiente:** Los hilos dentro de un proceso pueden comunicarse directamente mediante variables compartidas o mecanismos de sincronización de bajo nivel, como semáforos o mutex. En cambio, los procesos deben usar mecanismos de comunicación interprocesos (IPC) más costosos, como sockets, pipes o colas de mensajes.

**4. Mayor velocidad de creación y destrucción:** Crear y destruir hilos suele ser más rápido que crear y destruir procesos. Los hilos comparten recursos con el proceso principal, lo que significa que no se necesita un nuevo espacio de direcciones ni una copia de la imagen del proceso.

**5. Facilidad de coordinación:** Dado que los hilos comparten el mismo espacio de memoria, es más fácil coordinar y sincronizar la ejecución entre ellos. Por el contrario, los procesos deben utilizar mecanismos de sincronización más complejos y costosos, lo que puede afectar negativamente al rendimiento.

**4.3. ¿Cuáles son las dos características diferentes y potencialmente independientes en el concepto de proceso?**

Se muestra cómo el concepto de proceso es más complejo y sutil de lo que se ha visto hasta este momento y, de hecho, contiene dos conceptos diferentes y potencialmente independientes: uno relativo a la propiedad de recursos y otro relativo a la ejecución. En muchos sistemas operativos esta distinción ha llevado al desarrollo de estructuras conocidas como hilos (*threads*).

* **Propiedad de recursos.** Un proceso incluye un espacio de direcciones virtuales para el manejo de la imagen del proceso; como ya se explicó en el Capítulo 3 la imagen de un proceso es la colección de programa, datos, pila y atributos definidos en el bloque de control del proceso. De vez en cuando 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 realiza la función de protección para evitar interferencias no deseadas entre procesos en relación con los recursos.
* **Planificación/ejecución.** La ejecución de un proceso sigue una ruta de ejecución (traza) a través de uno o más programas. Esta ejecución puede estar intercalada con ese u otros procesos. De esta manera, un proceso tiene un estado de ejecución (Ejecutando, Listo, etc.) y una prioridad de activación y ésta es la entidad que se planifica y activa por el sistema operativo.

**4.4. Dé cuatro ejemplos generales del uso de hilos en un sistema multiprocesador monousuario.**

* **Trabajo en primer plano y en segundo plano.** Por ejemplo, en un programa de hoja de cálculo, un hilo podría mostrar menús y leer la entrada de usuario, mientras otro hilo ejecuta los mandatos de usuario y actualiza la hoja de cálculo. 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. Por ejemplo, se puede diseñar un procesador de textos con protección contra un fallo de corriente que escriba el *buffer* de su memoria RAM a disco una vez por minuto. Se puede crear un hilo cuyo único trabajo sea crear una copia de seguridad periódicamente y que se planifique directamente a través del sistema operativo; no se necesita código adicional en el programa principal que proporcione control de tiempo o que coordine la entrada/salida.
* **Velocidad de ejecución.** Un proceso multihilo puede computar una serie de datos mientras que lee los siguientes de un dispositivo. En un sistema multiprocesador pueden estar ejecutando simultáneamente múltiples hilos de un mismo proceso. De esta forma, aunque un hilo pueda estar bloqueado por una operación de E/S mientras lee datos, otro hilo puede estar ejecutando.
* **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.

**4.5. ¿Qué recursos son compartidos normalmente por todos los hilos de un proceso?**

Todos los hilos de un proceso comparten **el mismo espacio de direcciones de la memoria**, todos los hilos se suspenden al mismo tiempo. De forma similar, la finalización de un proceso finaliza todos los hilos de ese proceso.

El "*espacio de direcciones*" se refiere al conjunto de direcciones de memoria disponibles para un proceso o programa en un sistema computacional. Este espacio de direcciones representa la gama completa de ubicaciones de memoria que el proceso puede acceder durante su ejecución.

El espacio de direcciones se organiza en unidades de tamaño fijo, como bytes o palabras, y puede dividirse en varias secciones, cada una con un propósito específico.

**4.6. Enumere tres ventajas de los ULT sobre los KLT.**

Existen dos amplias categorías de implementación de hilos: hilos de nivel de usuario (*user-level threads*, ULT) e hilos de nivel de núcleo (*kernel-level threads,* KLT).

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.

**1.** **Hilos de nivel de usuario (ULT):**

- Los hilos de nivel de usuario son gestionados completamente por la biblioteca de hilos en el espacio de usuario del programa, sin intervención directa del sistema operativo.

- La creación, planificación y sincronización de los hilos de nivel de usuario se realizan mediante llamadas a funciones proporcionadas por la biblioteca de hilos, que generalmente se ejecutan en el espacio de usuario del programa.

- El sistema operativo no es consciente de la existencia de estos hilos y los trata como procesos normales. El núcleo del sistema operativo asigna recursos solo al proceso en el que se están ejecutando los hilos, y no tiene conocimiento ni control sobre los hilos individuales dentro del proceso.

**2. Hilos de nivel de núcleo (KLT):**

- Los hilos de nivel de núcleo, también conocidos como hilos soportados por el núcleo o procesos ligeros, son gestionados directamente por el núcleo del sistema operativo.

- El sistema operativo asigna recursos como el tiempo de CPU, la memoria y los recursos de E/S a cada hilo de nivel de núcleo, y gestiona su planificación, sincronización y cambio de contexto.

- Los hilos de nivel de núcleo generalmente requieren llamadas al sistema operativo para operaciones como la creación, planificación y sincronización de hilos, lo que puede resultar en un mayor overhead en comparación con los hilos de nivel de usuario.

- El sistema operativo trata cada hilo de nivel de núcleo como una entidad independiente y puede planificar y ejecutar los hilos de manera más flexible y eficiente, aprovechando las características del hardware subyacente.

**4.7. Enumere dos desventajas de los ULT en comparación con los KLT.**

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.

**4.8. Defina *jacketing (revestimiento)*.**

Otra forma de solucionar el problema de hilos que se bloquean es una técnica denominada *jacketing* (revestimiento). El objetivo de esta técnica es convertir una llamada al sistema bloqueante en una llamada al sistema no bloqueante. Por ejemplo, en lugar de llamar directamente a una rutina del sistema de E/S, un hilo puede llamar a una rutina *jacket* de E/S a nivel de aplicación. Con esta rutina *jacket*, el código verifica si el dispositivo de E/S está ocupado. Si lo está, el hilo entra en estado Bloqueado y pasa el control (a través de la biblioteca de hilos) a otro hilo. Cuando este hilo recupera de nuevo el control, chequea de nuevo el dispositivo de E/S.

**4.9. Defina brevemente las siguientes arquitecturas:**

**Procesadores paralelos:**

**- SIMD (única instrucción múltiples flujos de datos)**

**- MIMD (múltiples instrucciones múltiples flujos de datos):  
 - Memoria compartida (fuertemente acoplados):**

**- Maestro/esclavo**

**- Multiprocesadores simétricos (SMP)**

**- Memoria distribuida (débilmente acoplados:**

**- *Clusters***

**1. SIMD (Single Instruction, Multiple Data):**

- En SIMD, una sola instrucción es ejecutada simultáneamente en múltiples elementos de datos.

- Los procesadores SIMD son eficientes para aplicaciones que realizan operaciones idénticas en grandes conjuntos de datos, como procesamiento de imágenes, gráficos por computadora y procesamiento de señales.

**2. MIMD (Multiple Instruction, Multiple Data):**

- En MIMD, múltiples unidades de procesamiento ejecutan instrucciones independientes en conjuntos de datos independientes.

**a. Memoria compartida (fuertemente acoplados):**

- En sistemas con memoria compartida, varios procesadores comparten el mismo espacio de memoria física y pueden acceder a los mismos datos directamente.

- Ejemplos incluyen el modelo **maestro/esclavo**, donde un procesador coordina y distribuye tareas a los demás, y los sistemas **SMP**, donde todos los procesadores tienen acceso igualitario a la memoria.

**b. Memoria distribuida (débilmente acoplados):**

- En sistemas con memoria distribuida, cada procesador tiene su propio espacio de memoria local y se comunica con otros procesadores a través de mensajes.

- Estos sistemas pueden formar clústeres de computadoras interconectadas, como en el caso de los **clusters**, donde cada nodo tiene su propio procesador y memoria.

**4.10. Enumere los aspectos principales de diseño de un sistema operativo SMP.**

Las principales claves de diseño incluyen las siguientes características:

* **Procesos o hilos simultáneos concurrentes.** Las rutinas del núcleo necesitan ser reentrantes para permitir que varios procesadores ejecuten el mismo código del núcleo simultáneamente. Debido a que múltiples procesadores pueden ejecutar la misma o diferentes partes del código del núcleo, las tablas y la gestión de las estructuras del núcleo deben ser gestionas apropiadamente para impedir interbloqueos u operaciones inválidas.
* **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. En el Capítulo 10 se examina la planificación multiprocesador.

**~~4.11. Dé ejemplos de servicios y funciones que se encuentran en un sistema operativo monolítico típico que podrían ser subsistemas externos en un sistema operativo micronúcleo. 🡪 No estoy seguro~~** **No entra**

- **Sistema Operativo Monolítico:** En un sistema operativo monolítico, todas las funciones del sistema operativo se ejecutan en modo kernel en el mismo espacio de direcciones. Todas las funcionalidades, como la gestión de procesos, la gestión de memoria, el sistema de archivos y los controladores de dispositivos, forman parte de un solo y grande núcleo. Ejemplos de sistemas operativos monolíticos incluyen versiones antiguas de Linux y Windows (antes de Windows NT).

- **Sistema Operativo Micronúcleo:** En un sistema operativo micronúcleo, el núcleo es pequeño y solo proporciona servicios básicos como la comunicación entre procesos y la gestión de memoria. Funcionalidades como la gestión de procesos, el sistema de archivos y los controladores de dispositivos se implementan como subsistemas externos que se ejecutan en modo de usuario, fuera del núcleo. Ejemplos de sistemas operativos micronúcleo incluyen MINIX y QNX.

Servicios y funciones que podrían ser subsistemas externos en un sistema operativo micronúcleo:

**1. Gestión de Procesos:**

- En un sistema operativo monolítico, la gestión de procesos se realiza dentro del núcleo.

- En un sistema operativo micronúcleo, la gestión de procesos podría ser un subsistema externo que se ejecuta en modo de usuario. Este subsistema sería responsable de crear, destruir y gestionar procesos, así como de la planificación de la CPU.

**2. Sistema de Archivos:**

- En un sistema operativo monolítico, el sistema de archivos está integrado en el núcleo del sistema operativo.

- En un sistema operativo micronúcleo, el sistema de archivos podría ser un subsistema externo que se ejecuta en modo de usuario. Este subsistema sería responsable de proporcionar acceso a los archivos y directorios almacenados en los dispositivos de almacenamiento.

**3. Controladores de Dispositivos:**

- En un sistema operativo monolítico, los controladores de dispositivos están integrados en el núcleo del sistema operativo.

- En un sistema operativo micronúcleo, los controladores de dispositivos podrían ser subsistemas externos que se ejecutan en modo de usuario. Cada controlador de dispositivo sería responsable de interactuar con un dispositivo de hardware específico y proporcionar una interfaz para que otros componentes del sistema operativo accedan al dispositivo.

**4. Redes y Comunicaciones:**

- En un sistema operativo monolítico, las funciones de red y comunicaciones están integradas en el núcleo del sistema operativo.

- En un sistema operativo micronúcleo, las funciones de red y comunicaciones podrían ser subsistemas externos que se ejecutan en modo de usuario. Estos subsistemas serían responsables de proporcionar conectividad de red, protocolos de comunicación y servicios de red, como TCP/IP.

**4.12. Enumere y explique brevemente siete ventajas potenciales de un diseño micronúcleo en comparación con un diseño monolítico.**

**1. Interfaces uniformes:**

- El micronúcleo impone una interfaz uniforme en las peticiones realizadas por un proceso, ya que todos los servicios se proporcionan a través de paso de mensajes. Esto simplifica la interacción de los procesos con el sistema operativo.

**2. Extensibilidad:**

- La arquitectura de micronúcleo facilita la extensibilidad del sistema operativo, permitiendo agregar nuevos servicios o realizar múltiples servicios en la misma área funcional. Esto se logra mediante la adición o modificación de servidores relacionados sin necesidad de reconstruir todo el núcleo.

**3. Flexibilidad:**

- Las características existentes en un sistema de micronúcleo pueden ser eliminadas o modificadas para realizar una implementación más pequeña y eficiente. Esto permite que el sistema operativo sea flexible y se adapte a diferentes requisitos y entornos de uso.

**4. Portabilidad:**

- Un sistema operativo basado en un micronúcleo puede ser más portable, ya que gran parte del código específico del procesador está en el micronúcleo. Los cambios necesarios para migrar el sistema a un nuevo procesador son menores y tienden a estar unidos en grupos lógicos.

**5. Fiabilidad:**

- La estructura modular de un micronúcleo permite una mayor verificación y prueba rigurosa del código. Un micronúcleo pequeño puede ser verificado de manera más exhaustiva, lo que contribuye a mejorar la fiabilidad del sistema operativo.

**6. Soporte de sistemas distribuidos:**

- Un micronúcleo facilita el soporte de sistemas distribuidos, incluyendo clusters controlados por sistemas operativos distribuidos. La arquitectura de micronúcleo permite la comunicación entre procesos y servicios a través de paso de mensajes, lo que facilita la creación de sistemas distribuidos.

**7. Soporte de sistemas operativos orientados a objetos (OOOS):**

- Un micronúcleo es compatible con enfoques orientados a objetos en el diseño y desarrollo de extensiones modulares para el sistema operativo. Esto se logra mediante el uso de componentes con interfaces claramente definidas, que pueden ser interconectadas para construir el software a través de bloques de construcción.

**4.13. Explique la desventaja potencial de rendimiento de un sistema operativo 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.

Esta pérdida de rendimiento es más notable en los micronúcleos de primera generación, donde se observa una significativa reducción del rendimiento, incluso después de optimizar el código del micronúcleo. Para abordar este problema, se han explorado diferentes enfoques:

**1. \*\*Aumento del tamaño del micronúcleo\*\*:** Al reintroducir servicios críticos y manejadores en el micronúcleo, se reducen el número de cambios de modo usuario-núcleo y de espacio de direcciones de proceso. Sin embargo, esto puede sacrificar la fortaleza del diseño del micronúcleo, como sus mínimas interfaces y su flexibilidad.

**2. \*\*Reducción del tamaño del micronúcleo\*\*:** Diseñando un micronúcleo más pequeño, se eliminan las pérdidas de rendimiento y se mejora la flexibilidad y la fiabilidad. Ejemplos de micronúcleos pequeños de segunda generación incluyen L4, que consta de 12 Kbytes de código y 7 llamadas al sistema. Experimentaciones han demostrado que estos sistemas pueden funcionar tan bien o mejor que sistemas operativos monolíticos como UNIX.

**4.14. Enumere cuatro funciones que le gustaría encontrar incluso en un sistema operativo micronúcleo mínimo.**

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.

**4.15. ¿Cuál es la forma básica de comunicación entre procesos o hilos en un sistema operativo micronúcleo?**

La forma básica de comunicación entre procesos o hilos en un sistema operativo micronúcleo es a través de paso de mensajes (message passing). En este modelo de comunicación, los procesos o hilos intercambian datos y señales enviando y recibiendo mensajes entre sí. Esto se realiza típicamente a través de las siguientes operaciones básicas:

**1. Enviar mensaje:** Un proceso o hilo envía un mensaje a otro proceso o hilo especificando el destino y los datos que se desean transmitir. Esto puede implicar colocar los datos en un área de memoria compartida o estructuras de datos dedicadas para la comunicación.

**2. Recibir mensaje:** Un proceso o hilo espera y recibe mensajes que le son enviados por otros procesos o hilos. Para recibir un mensaje, el proceso o hilo especifica el remitente del mensaje y, en algunos casos, el tipo de mensaje que espera recibir.

**3. Responder a mensajes:** Después de recibir un mensaje, un proceso o hilo puede optar por responder al remitente enviando un mensaje de vuelta. Esto completa el ciclo de comunicación.

El paso de mensajes proporciona una forma flexible y segura de comunicación entre procesos o hilos en un sistema operativo micronúcleo. Permite la comunicación entre procesos o hilos que se ejecutan en diferentes espacios de direcciones y puede ser implementada de manera eficiente incluso en sistemas distribuidos. Además, el paso de mensajes puede ofrecer mecanismos de sincronización para garantizar la coordinación adecuada entre los procesos o hilos que participan en la comunicación.