Sistemas Operativos

Trabajo Prá ctico Gestión de memoria

**¯**

1. Problemas:
   1. **¿Qué parámetro del sistema se ve afectado directamente por la administración de memoria?**

El parámetro del sistema que se ve directamente afectado por la administración de memoria es el uso de la memoria física (RAM). La administración de memoria determina cómo se asignan, utilizan y liberan los recursos de memoria del sistema, lo que afecta directamente la cantidad de memoria disponible para las aplicaciones y el sistema operativo.

Una gestión eficiente de la memoria puede mejorar el rendimiento del sistema al garantizar que se utilice la memoria de manera óptima, mientras que una gestión deficiente puede llevar a problemas como la fragmentación de la memoria y la falta de recursos disponibles.

* 1. **¿Cuál es la diferencia entre una dirección física y una dirección virtual (dirección lógica)?**

La diferencia principal entre una dirección física y una dirección virtual (también llamada dirección lógica) radica en cómo se utilizan y gestionan en el contexto de la memoria de un sistema informático:

1. Dirección Física:

- Una dirección física es la dirección real en la memoria física del sistema. Representa una ubicación específica en los chips de memoria RAM.

- Las direcciones físicas son las que el hardware, como el controlador de memoria y la unidad de administración de memoria, utiliza para acceder directamente a los datos almacenados en la memoria física.

- Estas direcciones son únicas y están ligadas directamente a los chips de memoria física instalados en el sistema.

2. Dirección Virtual (Dirección Lógica):

- Una dirección virtual, o dirección lógica, es una dirección generada por el procesador y utilizada por el software (como los programas y el sistema operativo) para acceder a la memoria.

- Estas direcciones son relativas y se refieren a ubicaciones dentro del espacio de direcciones virtuales del proceso en ejecución.

- La dirección virtual se traduce en una dirección física a través de un proceso llamado mapeo de memoria, realizado por el sistema operativo y la unidad de administración de memoria.

- El uso de direcciones virtuales permite la implementación de técnicas como la memoria virtual, la segmentación y la paginación, que ofrecen flexibilidad y protección en la gestión de la memoria del sistema.

En resumen, mientras que las direcciones físicas son direcciones reales en la memoria física del sistema, las direcciones virtuales son direcciones generadas por el software y se traducen en direcciones físicas mediante la gestión de memoria del sistema operativo.

* 1. **Internamente ¿qué tipo de direcciones maneja la CPU?**

Internamente, la CPU maneja direcciones físicas y direcciones virtuales.

1. Direcciones Físicas:

- La CPU utiliza direcciones físicas para acceder directamente a la memoria física del sistema. Estas direcciones representan ubicaciones reales en los chips de memoria RAM.

- Cuando la CPU necesita acceder a datos o instrucciones en la memoria, emite direcciones físicas que son enviadas al controlador de memoria y utilizadas para recuperar los datos de la memoria física.

2. Direcciones Virtuales (Direcciones Lógicas):

- La CPU también maneja direcciones virtuales, que son generadas por el software en ejecución (programas y sistema operativo).

- Las direcciones virtuales se utilizan dentro de la CPU para acceder a la memoria de manera abstracta, sin tener en cuenta la ubicación física real de los datos.

- Estas direcciones virtuales son traducidas en direcciones físicas mediante el hardware de gestión de memoria, como la MMU (Memory Management Unit) en la CPU.

- La MMU realiza la traducción de direcciones virtuales a direcciones físicas utilizando tablas de páginas y otros mecanismos de mapeo de memoria, permitiendo que el software acceda a una memoria virtual más grande que la memoria física real disponible en el sistema.

En resumen, la CPU maneja tanto direcciones físicas como virtuales, utilizando direcciones físicas para acceder directamente a la memoria física y direcciones virtuales para interactuar con la memoria de manera abstracta, traduciéndolas en direcciones físicas cuando sea necesario mediante la MMU.

* 1. **¿Por qué el número de páginas y el taman˜o de las páginas es potencia de 2?**

El número de páginas y el tamaño de las páginas se eligen típicamente como potencias de 2 por varias razones:

1. Facilidad de Implementación:

- El uso de potencias de 2 simplifica muchos aspectos de la implementación del hardware y del software relacionados con la memoria virtual.

- Las operaciones de división y multiplicación por potencias de 2 pueden ser implementadas de manera más eficiente en el hardware y son más rápidas que las operaciones con números arbitrarios.

2. Alineación de Memoria:

- Muchos sistemas informáticos requieren que los datos estén alineados en la memoria para un acceso eficiente. Al tener tamaños de página y direcciones de página que son potencias de 2, se asegura que los datos estén alineados correctamente en la memoria, lo que puede mejorar el rendimiento de acceso.

3. Gestión de Memoria Física:

- Al utilizar tamaños de página que son potencias de 2, el hardware de gestión de memoria puede asignar y gestionar la memoria física de manera más eficiente.

- Por ejemplo, alinear las páginas con los límites de los bloques de memoria física facilita la asignación y liberación de páginas de memoria, y reduce la fragmentación de la memoria.

4. Optimización del Almacenamiento en Caché:

- Los tamaños de página potencia de 2 pueden coincidir con los tamaños de las cachés del hardware, lo que facilita la gestión del almacenamiento en caché y puede mejorar el rendimiento del sistema.

- Por ejemplo, si el tamaño de la caché es una potencia de 2 y el tamaño de la página coincide con el tamaño de la caché, es más fácil gestionar el almacenamiento en caché y minimizar los conflictos de caché.

En resumen, el uso de potencias de 2 para el número de páginas y el tamaño de las páginas simplifica la implementación del hardware y del software, mejora el rendimiento del sistema y facilita la gestión eficiente de la memoria.

* 1. **Explique la diferencia entre fragmentación interna y fragmentación externa.**

La fragmentación interna y externa son dos conceptos importantes en la gestión de la memoria que se refieren a diferentes tipos de desperdicio de recursos:

1. Fragmentación Interna:

- La fragmentación interna ocurre cuando un proceso recibe más memoria de la que realmente necesita. Como resultado, parte de la memoria asignada al proceso queda sin utilizar.

- Este tipo de fragmentación suele ocurrir en sistemas que asignan memoria en bloques fijos, como la asignación de memoria estática. Si un proceso solicita un bloque de memoria que es más grande que el tamaño del objeto que se almacenará, puede haber espacio no utilizado dentro del bloque asignado.

- La fragmentación interna es intrínseca al método de asignación de memoria utilizado y puede conducir a un desperdicio de recursos, especialmente en sistemas con asignación estática de memoria.

2. Fragmentación Externa:

- La fragmentación externa ocurre cuando hay suficiente memoria total disponible en el sistema, pero está distribuida en pequeños bloques dispersos que no pueden ser utilizados para satisfacer una solicitud de memoria contigua.

- Este tipo de fragmentación es más común en sistemas que utilizan asignación dinámica de memoria, como la asignación de memoria libre y la liberación de memoria. A medida que los procesos solicitan y liberan memoria, se crean huecos en la memoria que pueden ser demasiado pequeños para satisfacer futuras solicitudes de memoria.

- A diferencia de la fragmentación interna, la fragmentación externa afecta a la asignación de memoria dinámica y puede ser más difícil de gestionar, ya que los huecos dispersos pueden ser difíciles de encontrar y aprovechar de manera eficiente.

En resumen, la fragmentación interna se refiere al desperdicio de memoria dentro de bloques de asignación de memoria, mientras que la fragmentación externa se refiere al desperdicio de memoria debido a la distribución no contigua de bloques de memoria en el sistema. Ambos tipos de fragmentación pueden reducir la eficiencia y el rendimiento del sistema si no se gestionan adecuadamente.

* 1. **¿Qué ventajas tiene la utilización de p´aginas multinivel?**

La utilización de páginas multinivel, también conocida como paginación multinivel, ofrece varias ventajas en comparación con los sistemas de paginación de un solo nivel:

1. Reducción de la Tabla de Páginas Principal:

- En un sistema de paginación de un solo nivel, la tabla de páginas principal debe contener una entrada para cada página de memoria virtual, lo que puede ser prohibitivo en sistemas con una gran cantidad de memoria.

- Con la paginación multinivel, la tabla de páginas principal se divide en varias tablas de páginas más pequeñas, lo que reduce la cantidad de entradas necesarias en cada tabla y, por lo tanto, la cantidad total de memoria necesaria para almacenar la tabla de páginas.

2. Eficiencia en la Búsqueda de Páginas:

- En un sistema de paginación multinivel, la búsqueda de una página en la tabla de páginas se realiza en etapas. La primera etapa implica la búsqueda en la tabla de páginas de nivel superior, y si se encuentra la entrada correspondiente, se accede a la tabla de páginas de nivel inferior para encontrar la página específica.

- Este enfoque de búsqueda en etapas puede ser más eficiente que la búsqueda lineal en una tabla de páginas única, especialmente en sistemas con una gran cantidad de páginas.

3. Gestión de la Memoria Física:

- La paginación multinivel puede facilitar la gestión de la memoria física al permitir que diferentes niveles de la tabla de páginas se asignen en ubicaciones de memoria física separadas.

- Esto puede ayudar a reducir la fragmentación de la memoria física y mejorar la eficiencia de la gestión de la memoria.

4. Soporte para Espacios de Direcciones Virtuales Grandes:

- La paginación multinivel es especialmente útil en sistemas con espacios de direcciones virtuales grandes, ya que permite dividir la tabla de páginas en estructuras más manejables y escalables.

- Esto facilita la gestión de la memoria en sistemas con una gran cantidad de procesos en ejecución y/o grandes espacios de direcciones virtuales.

En resumen, la paginación multinivel proporciona una manera eficiente y escalable de gestionar la memoria virtual en sistemas informáticos modernos, reduciendo la sobrecarga de memoria necesaria para almacenar la tabla de páginas, mejorando la eficiencia en la búsqueda de páginas y facilitando la gestión de la memoria física.

* 1. **Analice los modelos de memoria: paginada, segmentada y segmentación con página- ci´on. Discuta las ventajas y desventajas de cada uno de ellos.**

1. Memoria Paginada:

- Ventajas:

- Elimina la fragmentación externa: Al dividir la memoria en páginas de tamaño fijo, se evita la fragmentación externa, ya que los procesos pueden asignarse páginas contiguas sin importar su ubicación física.

- Uso eficiente de la memoria: Permite una gestión eficiente de la memoria al asignar solo las páginas necesarias para un proceso, lo que puede reducir el desperdicio de memoria.

- Facilita la gestión de la memoria: Simplifica la gestión de la memoria al permitir que el sistema operativo maneje las páginas de forma independiente, lo que facilita la implementación de técnicas como la paginación anticipada y la paginación bajo demanda.

- Desventajas:

- Fragmentación interna: Puede haber desperdicio de memoria dentro de las páginas, especialmente si los procesos no utilizan todo el espacio de una página.

- Sobrecarga del hardware: La gestión de las tablas de páginas y la traducción de direcciones pueden introducir una sobrecarga en el hardware, especialmente en sistemas con grandes espacios de direcciones virtuales.

2. Memoria Segmentada:

- Ventajas:

- Soporte para espacios de direcciones de tamaño variable: Permite dividir el espacio de direcciones en segmentos de diferentes tamaños, lo que facilita la gestión de procesos con requisitos de memoria heterogéneos.

- Protección y compartición de memoria: Los segmentos pueden asignarse permisos de acceso diferentes, lo que permite implementar esquemas de protección y compartición de memoria más flexibles.

- Desventajas:

- Fragmentación interna y externa: Puede haber fragmentación interna si los segmentos no se llenan por completo y fragmentación externa si los segmentos no están contiguos en memoria.

- Gestión compleja: La gestión de segmentos puede ser más compleja que la paginación, ya que los segmentos pueden crecer y contraerse dinámicamente, lo que requiere una gestión más sofisticada por parte del sistema operativo.

3. Segmentación con Paginación:

- Ventajas:

- Combina las ventajas de la segmentación y la paginación: Permite gestionar eficientemente la memoria utilizando segmentos para organizar lógicamente el espacio de direcciones y páginas para administrar la memoria física.

- Flexibilidad: Proporciona la flexibilidad de la segmentación para manejar espacios de direcciones de tamaño variable y la eficiencia de la paginación para evitar la fragmentación externa.

- Desventajas:

- Complejidad: La combinación de segmentación y paginación puede introducir una mayor complejidad en la gestión de la memoria y en la traducción de direcciones.

- Sobrecarga de hardware: Requiere hardware adicional para gestionar tanto la segmentación como la paginación, lo que puede aumentar la complejidad y el costo del sistema.

En resumen, cada modelo de memoria tiene sus propias ventajas y desventajas, y la elección entre ellos depende de los requisitos específicos del sistema y las aplicaciones que se ejecutarán en él. La paginación se destaca por su simplicidad y eficiencia en la gestión de la memoria física, mientras que la segmentación ofrece flexibilidad en la gestión del espacio de direcciones. La segmentación con paginación combina lo mejor de ambos mundos, pero puede ser más compleja de implementar y gestionar.

* 1. **¿Cómo se protegen los esquemas de asignación del ejercicio 7?**

En el ejercicio 7, se utilizan esquemas de asignación de memoria como la paginación multinivel y la segmentación. Estos esquemas pueden incluir mecanismos de protección para garantizar la seguridad y la integridad del sistema. A continuación, describiré cómo se pueden proteger estos esquemas:

1. Paginación Multinivel:

- Protección por permisos de página: Cada página de memoria puede tener asociados permisos de acceso que determinan si un proceso puede leer, escribir o ejecutar esa página. Estos permisos se gestionan a nivel de página y se almacenan en las tablas de páginas.

- Protección de tablas de páginas: Las tablas de páginas también pueden estar protegidas por permisos para evitar la manipulación no autorizada. Solo el sistema operativo debe tener permisos para modificar las tablas de páginas.

- Protección del espacio de direcciones: Cada proceso tiene su propio espacio de direcciones virtuales, lo que proporciona aislamiento entre los procesos. Esto impide que un proceso acceda a la memoria de otro proceso sin los permisos adecuados.

- Protección de la memoria física: El hardware puede proporcionar mecanismos de protección para evitar que un proceso acceda a áreas de memoria física que no le corresponden. Esto se logra mediante la asignación de permisos de acceso a las páginas de memoria física.

2. Segmentación:

- Protección por permisos de segmento: Cada segmento de memoria puede tener permisos asociados que determinan qué tipo de acceso está permitido (lectura, escritura, ejecución). Estos permisos se gestionan a nivel de segmento y se almacenan en las tablas de segmentos.

- Protección de tablas de segmentos: Al igual que con la paginación, las tablas de segmentos también pueden estar protegidas para evitar la manipulación no autorizada. Solo el sistema operativo debe tener permisos para modificar las tablas de segmentos.

- Protección del espacio de direcciones: Cada proceso tiene su propio conjunto de segmentos de memoria, lo que proporciona aislamiento entre los procesos y evita que un proceso acceda a la memoria de otro proceso sin los permisos adecuados.

- Protección de la memoria física: El hardware puede proporcionar mecanismos de protección similares a los de la paginación para evitar accesos no autorizados a la memoria física.

En resumen, tanto la paginación multinivel como la segmentación pueden protegerse mediante permisos de acceso a nivel de página o segmento, protección de las tablas de páginas o segmentos, aislamiento entre los espacios de direcciones de los procesos y mecanismos de protección de la memoria física. Estos mecanismos garantizan que solo los procesos autorizados puedan acceder a la memoria y que se mantenga la integridad del sistema.

* 1. **La mayoría de los sistemas permiten a los programas asignar más memoria a su espacio de direcciones durante la ejecución. Como ejemplo de ese tipo de asignación de memoria tenemos los datos asignados en los segmentos de los programas dedicados a cumulo de memoria. ¿Qué se necesitaría para soportar la asignación dinámica de memoria en los siguientes esquemas?**
     1. **asignación contigua de memoria**
     2. **segmentación pura**
     3. **paginación pura**

Para soportar la asignación dinámica de memoria en los siguientes esquemas, se necesitarían ciertos mecanismos y técnicas específicas:

a) Asignación Contigua de Memoria:

- Algoritmos de asignación de memoria dinámica: Se requerirían algoritmos eficientes para asignar y liberar bloques de memoria de manera dinámica dentro del espacio de direcciones contiguo. Algunos ejemplos incluyen algoritmos como el primer ajuste, el mejor ajuste o el peor ajuste.

- Gestión de la fragmentación: Debería haber mecanismos para gestionar la fragmentación interna y externa que pueda ocurrir con la asignación dinámica de memoria. Esto podría implicar técnicas como la compactación de memoria para reducir la fragmentación.

- Protección de la memoria: Se necesitarían mecanismos para garantizar la protección de la memoria asignada, como permisos de acceso y segmentación de la memoria asignada a diferentes procesos.

b) Segmentación Pura:

- Tabla de segmentos dinámica: Sería necesario implementar una tabla de segmentos dinámica que pueda crecer o reducirse según las necesidades de asignación de memoria de los procesos en ejecución.

- Gestión de la fragmentación: Al igual que en la asignación contigua de memoria, se requerirían técnicas para gestionar la fragmentación de los segmentos, como la compactación de segmentos o la reubicación dinámica.

- Protección de la memoria: Se necesitarían mecanismos para garantizar la protección de los segmentos asignados, como permisos de acceso y gestión de los límites de los segmentos para evitar accesos no autorizados.

c) Paginación Pura:

- Tabla de páginas dinámica: Debería implementarse una tabla de páginas dinámica que pueda crecer o reducirse según las necesidades de asignación de memoria de los procesos en ejecución.

- Gestión del espacio de direcciones virtuales: Se requeriría un mecanismo para gestionar el espacio de direcciones virtuales de manera dinámica, asignando y liberando páginas de memoria según sea necesario.

- Gestión de la fragmentación: La paginación puede ayudar a reducir la fragmentación externa, pero aún puede ocurrir fragmentación interna en las páginas. Se podrían implementar técnicas de compactación de páginas para reducir la fragmentación interna.

- Protección de la memoria: Se necesitarían mecanismos para garantizar la protección de las páginas asignadas, como permisos de acceso y gestión de los bits de protección en las entradas de la tabla de páginas.

En resumen, para admitir la asignación dinámica de memoria en estos esquemas, se necesitarían adaptaciones específicas en la gestión de la memoria, incluida la implementación de estructuras de datos dinámicas, algoritmos de asignación eficientes y mecanismos de protección adecuados.

* 1. **Analice que ventajas y desventajas se obtiene de la utilización de páginas grandes y de páginas pequeñas.**

Utilizar páginas grandes y páginas pequeñas en la gestión de memoria tiene sus ventajas y desventajas:

Páginas Grandes:

Ventajas:

1. Reducción de la sobrecarga del sistema: Con páginas grandes, se reduce la cantidad de entradas en las estructuras de datos de administración de la memoria, como las tablas de páginas. Esto puede disminuir la sobrecarga del sistema y mejorar el rendimiento general.

2. Menor consumo de recursos: Al tener menos entradas en las estructuras de datos de administración de la memoria, se reduce el consumo de recursos del sistema, como la memoria y el tiempo de CPU necesario para administrar las páginas.

3. Menor fragmentación interna: Con páginas grandes, se reduce la fragmentación interna de la memoria, ya que cada página contiene más datos y es menos probable que queden espacios no utilizados dentro de una página.

Desventajas:

1. Mayor fragmentación externa: Las páginas grandes pueden aumentar la fragmentación externa, ya que es posible que se asignen páginas enteras incluso si solo se necesita una pequeña porción de la página. Esto puede llevar a un uso ineficiente de la memoria.

2. Requisitos de asignación de memoria: La asignación de memoria con páginas grandes puede ser menos flexible, ya que se requiere asignar bloques de tamaño completo de página, incluso si el proceso no necesita toda la página. Esto puede conducir a un uso ineficiente de la memoria.

3. Mayor tiempo de inicialización: El tiempo de inicialización del sistema puede ser mayor con páginas grandes, ya que se deben inicializar y asignar menos páginas grandes en comparación con un mayor número de páginas pequeñas.

Páginas Pequeñas:

Ventajas:

1. Mayor flexibilidad en la asignación de memoria: Las páginas pequeñas permiten una asignación más granular de la memoria, lo que significa que los procesos pueden utilizar solo la cantidad de memoria que necesitan realmente, sin desperdiciar espacio.

2. Menor fragmentación externa: Con páginas pequeñas, es menos probable que se desperdicie memoria debido a la fragmentación externa, ya que es más probable que se asignen solo las porciones necesarias de una página.

3. Mejor utilización de la memoria: Al permitir una asignación más precisa de la memoria, las páginas pequeñas pueden conducir a una mejor utilización general de la memoria del sistema.

Desventajas:

1. Mayor sobrecarga del sistema: Con páginas pequeñas, se requieren más entradas en las estructuras de datos de administración de la memoria, lo que puede aumentar la sobrecarga del sistema y reducir el rendimiento.

2. Mayor consumo de recursos: Al tener más entradas en las estructuras de datos de administración de la memoria, se requiere más memoria y tiempo de CPU para administrar las páginas pequeñas.

3. Mayor fragmentación interna: Con páginas pequeñas, es posible que se produzca una fragmentación interna más significativa, ya que los procesos pueden ocupar solo una fracción de una página, dejando espacio no utilizado dentro de la página.

En resumen, la elección entre páginas grandes y páginas pequeñas depende de diversos factores, como las necesidades específicas del sistema, el rendimiento, la eficiencia en el uso de la memoria y la gestión de la fragmentación. Es importante considerar estas ventajas y desventajas al diseñar y configurar un sistema de gestión de memoria.

* 1. **El diagrama de la figura 1 muestra un ejemplo del estado de memoria utilizando arma- nitración de alocaci´on contigua dinámica, después de varias operaciones de reemplazo y swapping-out ocurridas en el sistema. Las direcciones van de izquierda a derecha.**

****

**Figura 1: Configuraci´on de memoria**

**Las ´areas grises representan a procesos que est´an cargados en memoria; y las ´areas blancas indican bloques de memoria libre. El u´ltimo proceso cargado en memoria requiri´o 2 Mbytes y est´a marcado con una X. Solamente una operación de swap-out ocurri´o despu´es de la u´ltima carga.**

* + 1. **¿Cu´al fue el taman˜o m´aximo del proceso intercambiado?**
    2. **¿Cu´al era el taman˜o del bloque libre justo antes de que X lo particionara?**
    3. **Una nueva solicitud de asignaci´on de 3 Mbytes es requerida. Indicar los bloques de memoria donde se crear´a una partici´on para el nuevo proceso considerando los cuatro algoritmos de alocaci´on din´amica: mejor ajuste, primer ajuste, siguiente ajuste y peor ajuste.**

1. Para determinar el tamaño máximo del proceso intercambiado, observamos el área más grande de memoria contigua que se liberó antes de que se cargara el proceso X. En la imagen proporcionada, el área más grande de memoria libre es de 3 MB (el área gris justo antes de la carga del proceso X). Por lo tanto, el tamaño máximo del proceso intercambiado fue de 3 MB.
2. El tamaño del bloque libre justo antes de que X lo particionara fue de 2 MB (el área blanca a la derecha del proceso X).
3. Para la nueva solicitud de asignación de 3 MB, indicaremos los bloques de memoria donde se creará una partición para el nuevo proceso considerando los cuatro algoritmos de asignación dinámica:

- Mejor ajuste: Se selecciona el bloque de memoria libre más pequeño que sea lo suficientemente grande como para alojar el proceso. En este caso, se seleccionaría el bloque de 3 MB, ya que es el único que puede acomodar la solicitud de 3 MB.

- Primer ajuste: Se selecciona el primer bloque de memoria libre lo suficientemente grande para alojar el proceso. En este caso, también se seleccionaría el bloque de 3 MB.

- Siguiente ajuste: Similar al primer ajuste, pero comienza la búsqueda desde el bloque donde se alojó el último proceso. En este caso, dado que solo hay un bloque libre después de la última carga, se seleccionaría ese bloque de 3 MB.

- Peor ajuste: Se selecciona el bloque de memoria libre más grande disponible. En este caso, nuevamente se seleccionaría el bloque de 3 MB, ya que es el único disponible.

Por lo tanto, en todos los casos, el nuevo proceso se asignaría al único bloque de memoria libre disponible, que tiene un tamaño de 3 MB.

* 1. **En un sistema con paginación, un proceso no puede acceder a una zona de memoria que no sea de su propiedad. ¿Por qué? ¿Cómo podría el sistema operativo permitir el acceso a otras zonas de memoria? ¿Por qu´e debería o por qu´e no debería?**

En un sistema con paginación, cada proceso tiene su propia tabla de páginas que mapea las direcciones virtuales a las direcciones físicas de la memoria. Esta tabla de páginas se utiliza para controlar y supervisar el acceso a la memoria.

1. ¿Por qué un proceso no puede acceder a una zona de memoria que no es de su propiedad?

Esto se debe a que el sistema operativo utiliza el mecanismo de protección de memoria para garantizar la seguridad y la integridad de los procesos. Cada proceso tiene su propia tabla de páginas que indica qué páginas de memoria le pertenecen. Si un proceso intenta acceder a una página que no le pertenece, el hardware de la CPU generará una excepción de violación de segmento o página, lo que detendrá el acceso no autorizado.

2. ¿Cómo podría el sistema operativo permitir el acceso a otras zonas de memoria?

El sistema operativo podría permitir el acceso a otras zonas de memoria mediante técnicas como compartir memoria entre procesos (por ejemplo, mediante el uso de mecanismos de comunicación interproceso como pipes, colas, memoria compartida) o mediante permisos especiales otorgados por el sistema operativo.

3. ¿Por qué debería o por qué no debería permitir el acceso a otras zonas de memoria?

Permitir el acceso a otras zonas de memoria podría ser beneficioso en ciertos casos, como en entornos de multiprogramación donde se requiere compartir datos entre procesos o para mejorar la eficiencia de la memoria. Sin embargo, también conlleva riesgos de seguridad, ya que podría permitir que un proceso malicioso acceda a la memoria de otros procesos y comprometa la seguridad del sistema. Por lo tanto, el sistema operativo debe evaluar cuidadosamente los riesgos y beneficios antes de permitir el acceso a otras zonas de memoria, y debe implementar mecanismos de protección adecuados para garantizar la seguridad y la integridad del sistema.

* 1. **Considere un sistema de paginación en el que la tabla de páginas est´e almacenada en memoria.**
     1. **Si una referencia a memoria tarda en realizarse 200 nanosegundos, ¿cuánto tiempo tardar´a una referencia a memoria paginada?**
     2. **Si añadimos buffers TLB y el 75 % de todas las referencias a las tablas de páginas se encuentran en los buffers TLB, ¿cuál es el tiempo efectivo que tarda una referencia a memoria? (Suponga que la localización a una entrada de la tabla de páginas contenida en los buffers TLB se hace en tiempo cero, si la entrada ya se encuentra allí).**

1. Si una referencia a memoria tarda 200 nanosegundos y consideramos un sistema de paginación donde la tabla de páginas está almacenada en memoria, entonces cada referencia a memoria paginada debe pasar por dos accesos a memoria: uno para acceder a la tabla de páginas y otro para acceder a la página real en la memoria física.

Suponiendo que el acceso a la tabla de páginas y la página real en la memoria física toman el mismo tiempo (200 nanosegundos cada uno), entonces el tiempo total para una referencia a memoria paginada sería de 400 nanosegundos.

1. Si el 75% de todas las referencias a las tablas de páginas se encuentran en los buffers TLB (Translation Lookaside Buffer), entonces solo el 25% de las referencias a las tablas de páginas requerirán un acceso a memoria adicional para buscar la entrada en la tabla de páginas.

Dado que el acceso a una entrada en el TLB es instantáneo (tiempo cero), el tiempo efectivo que tarda una referencia a memoria sería el tiempo necesario para acceder al TLB más el tiempo necesario para acceder a la página real en la memoria física para el 25% de las referencias, más el tiempo total para acceder a la tabla de páginas y la página real en la memoria física para el restante 75% de las referencias.

Por lo tanto, el tiempo efectivo se calcularía de la siguiente manera:

Tiempo efectivo = (0.75 x 200 ns) + (0.25 x (200 ns + 200 ns))

Tiempo efectivo = (150 ns) + (0.25 x 400 ns)

Tiempo efectivo = 150 ns + 100 ns = 250 ns

Entonces, el tiempo efectivo que tarda una referencia a memoria en este caso sería de 250 nanosegundos.

* 1. **¿En qu´e caso utilizar´ıa p´agina invertida? ¿Por qu´e?**

La paginación invertida es una técnica en la gestión de memoria virtual donde en lugar de mantener una tabla de páginas por proceso, se mantiene una sola tabla de páginas para todo el sistema operativo. Cada entrada de esta tabla apunta a una ubicación en el disco donde se encuentra almacenada la página correspondiente. Esta técnica se utiliza en sistemas con gran cantidad de procesos y/o con una cantidad limitada de memoria física.

Se podría considerar el uso de la paginación invertida en los siguientes casos:

1. Escasez de memoria física: Cuando el sistema tiene una cantidad limitada de memoria física y debe manejar múltiples procesos que requieren más memoria de la disponible. En este escenario, la paginación invertida permite un uso más eficiente de la memoria física al permitir que las páginas de memoria no utilizadas se almacenen en el disco y se traigan de vuelta cuando sea necesario.

2. Multiprogramación: En entornos donde se ejecutan múltiples procesos simultáneamente y la cantidad total de memoria necesaria para todos los procesos excede la cantidad de memoria física disponible. La paginación invertida permite administrar de manera más eficiente la memoria virtual de todos los procesos en el sistema, asignando y desasignando páginas según sea necesario.

3. Fragmentación externa: En sistemas donde la fragmentación externa es un problema significativo, la paginación invertida puede ayudar a mitigar este problema al permitir que las páginas se almacenen de manera más compacta en el disco, liberando espacio en la memoria física para nuevas asignaciones.

En resumen, la paginación invertida es útil en entornos donde la eficiencia en el uso de la memoria física es crítica y donde se necesita manejar de manera efectiva una gran cantidad de procesos con una cantidad limitada de recursos de memoria física.

* 1. **Describa las acciones que realiza un sistema operativo cuando ocurre page fault.**

Cuando ocurre un fallo de página (page fault) en un sistema operativo con gestión de memoria paginada, significa que el proceso ha intentado acceder a una página de memoria que no se encuentra actualmente en la memoria física, sino que está almacenada en el disco. Ante un fallo de página, el sistema operativo realiza las siguientes acciones:

1. Interrupción del proceso: Cuando se detecta un fallo de página, se interrumpe temporalmente la ejecución del proceso que intentó acceder a la página no presente en memoria.

2. Búsqueda en la tabla de páginas: El sistema operativo busca en la tabla de páginas del proceso la entrada correspondiente a la página que causó el fallo. Esta entrada contiene información sobre la ubicación de la página en el disco.

3. Selección de una página a reemplazar: Si no hay suficiente espacio en la memoria física para cargar la página desde el disco, el sistema operativo debe seleccionar una página para reemplazarla. Esto puede implicar la aplicación de un algoritmo de reemplazo de página, como el algoritmo de reemplazo FIFO (primeras entradas, primeras salidas), LRU (menos recientemente usado), etc.

4. Lectura desde el disco: Una vez seleccionada la página a reemplazar, el sistema operativo realiza una operación de E/S (entrada/salida) para leer la página requerida desde el disco al espacio de la memoria física.

5. Actualización de la tabla de páginas: Se actualiza la entrada correspondiente en la tabla de páginas del proceso para reflejar que la página ahora está presente en la memoria física.

6. Reanudación del proceso: Finalmente, se reanuda la ejecución del proceso que experimentó el fallo de página. El acceso a la página que causó el fallo ahora se completará con éxito, ya que la página está disponible en la memoria física.

Estas acciones son parte del manejo de fallos de página del sistema operativo, que permite que los procesos accedan a páginas de memoria de manera transparente, independientemente de si están presentes en la memoria física o almacenadas en el disco.

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**