**Practica 5**

***Dirección Lógica o Virtual***

Es una dirección que enmascara o abstrae una dirección física. Referencia a una localidad en memoria y se la debe traducir a una dirección física.

***Dirección Física***

Es la dirección real. Es con la que se accede efectivamente a memoria***.*** Representa la dirección absoluta en memoria principal.

***Particiones Fijas***

La memoria se divide en particiones o regiones de tamaño fijo → tamaños iguales o diferentes. Alojan un único proceso. Cada proceso se coloca en alguna partición de acuerdo a algún criterio:

* First Fit
* Best Fit
* Worst Fit
* Next Fit

Diferencias:

* + Tamaño de Partición Fijo: La principal diferencia es que las particiones tienen tamaños fijos. Esto significa que, si un proceso es más pequeño que la partición asignada, se desperdicia espacio.
  + Fragmentación Interna: Puede haber fragmentación interna en las particiones fijas, lo que es el espacio no utilizado dentro de una partición.

Ventajas:

* + Simplicidad: Es más fácil de implementar y administrar, ya que no se requiere un seguimiento constante de la memoria libre.
  + Predicción: Los procesos pueden prever el espacio que se les asignará.

Desventajas:

* + Desperdicio de Espacio: Puede haber un desperdicio de espacio debido a la fragmentación interna.
  + Limitaciones para Procesos Grandes o Pequeños: Los procesos deben ajustarse al tamaño de la partición, lo que puede ser problemático si hay procesos muy grandes o pequeños.

***Particiones Dinámicas***

Las particiones varían en tamaño y numero. Alojan un proceso cada una. Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso.

Diferencias:

* + Tamaño de Partición Variable: Las particiones pueden ajustarse al tamaño de los procesos, lo que reduce el desperdicio de espacio.
  + Fragmentación Externa: En lugar de la fragmentación interna, este método puede sufrir de fragmentación externa, que es la ocupación de pequeños espacios libres en medio de particiones.

Ventajas:

* + Uso Eficiente de la Memoria: Reduce el desperdicio de espacio debido a la asignación más flexible de particiones.
  + Acomodación de Procesos de Diferentes Tamaños: Puede manejar procesos de tamaños diversos de manera más efectiva.

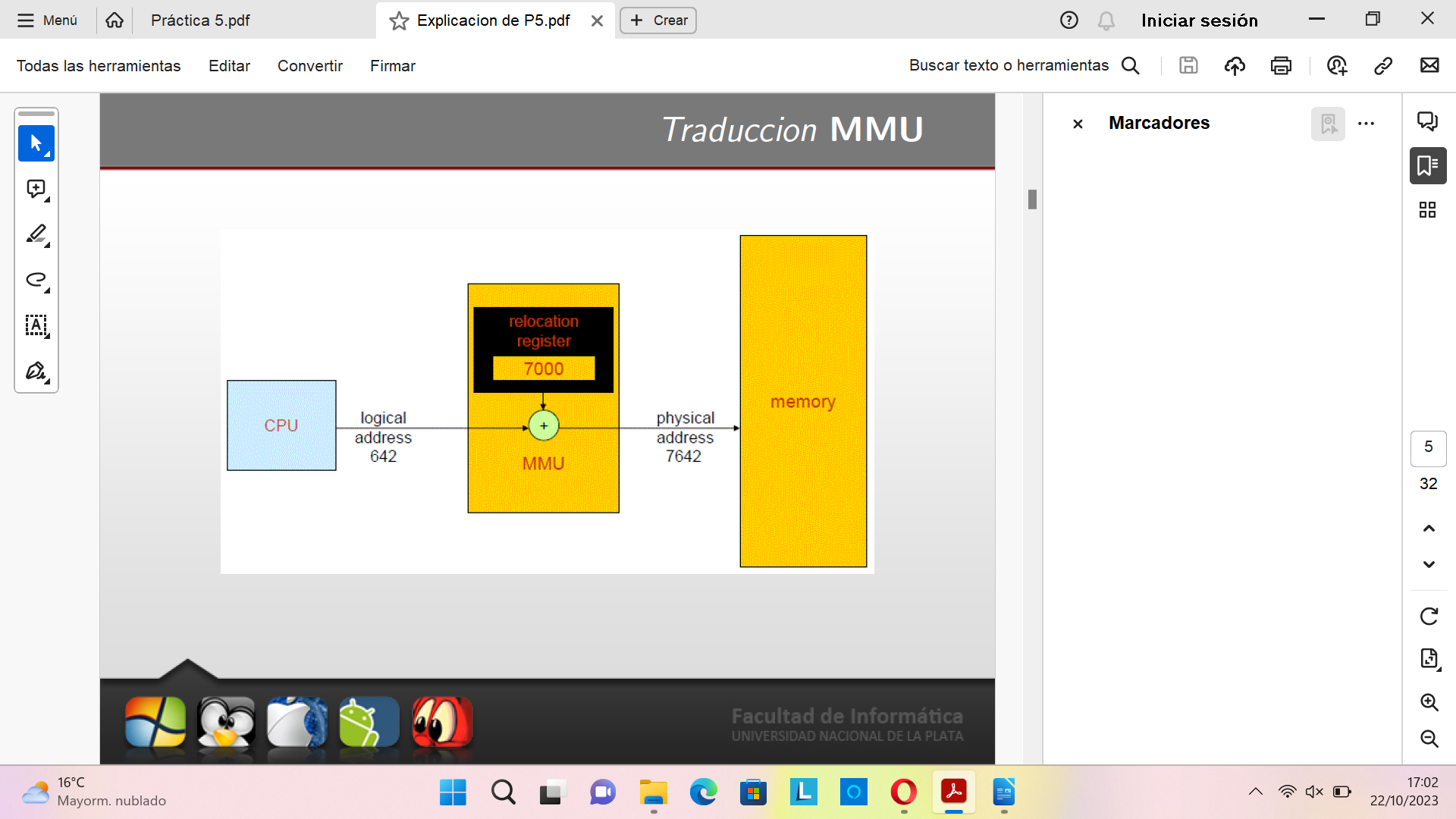
Desventajas:

* + Complejidad de Implementación: Requiere una gestión más compleja para asignar y liberar particiones de memoria.
  + Fragmentación Externa: Puede haber problemas de fragmentación externa si no se administra adecuadamente.

***Que información debe disponer el SO para poder administrar la memoria con estos métodos?***

Para que un sistema operativo pueda administrar la memoria con los métodos de Particiones Múltiples, ya sea con Particiones Fijas o Particiones Dinámicas, necesita disponer de cierta información y estructuras de datos para llevar a cabo una gestión efectiva de la memoria. Aquí hay información que el sistema operativo debe tener:

* Tabla de Particiones: Tanto en el caso de Particiones Fijas como en el de Particiones Dinámicas, el sistema operativo necesita una tabla que registre la información sobre cada partición de memoria. Esta tabla incluye detalles como la dirección de inicio, el tamaño, el estado (ocupada o libre) y el proceso al que se ha asignado si es el caso.
* Tabla de Procesos: Para la administración de la memoria con Particiones Múltiples, el sistema operativo debe llevar un registro de todos los procesos en ejecución. Esta tabla incluye información sobre cada proceso, como su identificador (PID), el tamaño del proceso y su estado (activo, en espera, suspendido, etc.).
* Algoritmos de Asignación de Particiones: El sistema operativo necesita algoritmos de asignación que determinen cuál es la partición de memoria adecuada para cargar un proceso. Esto implica tener una lógica que evalúe si una partición está libre, si es lo suficientemente grande para el proceso y cómo se realiza la asignación.
* Algoritmos de Liberación de Memoria: Para liberar memoria una vez que un proceso ha terminado o ha sido suspendido, el sistema operativo debe contar con algoritmos para administrar la liberación de particiones de memoria y su marcado como libres.
* Gestión de Fragmentación: En el caso de Particiones Múltiples, el sistema operativo debe gestionar la fragmentación, que puede ser interna (en Particiones Fijas) o externa (en Particiones Dinámicas). Esto implica estrategias para reducir la fragmentación, como compactación o reubicación de procesos.
* Registros de Memoria Libre: Para Particiones Dinámicas, es necesario llevar un registro de los bloques de memoria libres disponibles. Esto facilita la asignación de particiones a procesos.
* Mecanismos de Protección y Aislamiento: El sistema operativo también debe contar con mecanismos para asegurar que un proceso no acceda a la memoria de otros procesos. Esto implica establecer límites y permisos para cada partición de memoria asignada.
* Mecanismos de Intercambio (Swapping) y Paginación: En sistemas más avanzados, es posible que se utilicen mecanismos de intercambio o paginación para administrar la memoria de manera más eficiente. Estos métodos implican la gestión de áreas de almacenamiento secundario y su relación con la memoria principal.

***Grafico indicado como realiza el SO la transformación de direcciones lógicas a direcciones físicas.***

Particiones de Igual Tamaño

Ventajas

* Simplicidad: Este enfoque es simple de implementar y administrar, ya que todas las particiones tienen el mismo tamaño. No se necesita un seguimiento detallado del tamaño de cada partición.
* Predicción: Los procesos pueden prever el espacio que se les asignará, lo que facilita la planificación de la memoria.

Desventajas

* Desperdicio de Espacio: El principal inconveniente de las particiones de igual tamaño es que pueden resultar en un desperdicio significativo de espacio, ya que los procesos deben ajustarse al tamaño de la partición, lo que puede dejar espacio sin utilizar en particiones más pequeñas.
* Limitaciones para Procesos Grandes o Pequeños: Los procesos deben ser lo suficientemente pequeños como para caber en una partición o se producirá un desperdicio de espacio o, en el caso de procesos grandes, se requerirán múltiples particiones.

*Particiones de Diferente Tamaño*

Ventajas

* Uso Eficiente de la Memoria: Las particiones de diferente tamaño permiten un uso más eficiente de la memoria, ya que los procesos pueden asignarse a particiones que se ajusten mejor a su tamaño, reduciendo así el desperdicio de espacio.
* Flexibilidad: Se pueden acomodar tanto procesos pequeños como grandes, lo que aumenta la flexibilidad en la asignación de memoria.

Desventajas

* Complejidad de Implementación: La administración de particiones de diferente tamaño es más compleja que las particiones de igual tamaño. Requiere llevar un registro detallado del estado de cada partición y asignar procesos de manera más dinámica.
* Fragmentación Externa: En las particiones de diferente tamaño, es más probable que se produzca fragmentación externa, ya que los espacios libres entre las particiones pueden no ser lo suficientemente grandes para acomodar un proceso nuevo.

En resumen, las particiones de igual tamaño son más simples de administrar, pero pueden resultar en un mayor desperdicio de espacio y limitaciones para procesos de diferentes tamaños. Las particiones de diferente tamaño ofrecen un uso más eficiente de la memoria y mayor flexibilidad, pero son más complejas de implementar y pueden experimentar fragmentación externa. La elección entre estos dos métodos dependerá de las necesidades y restricciones específicas del sistema operativo y las aplicaciones que se ejecuten en él.

***Fragmentación***

La fragmentación se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua.

***Fragmentación Interna (Para el caso de Particiones Fijas)***

Es interna a la localidad asignada *y es la porción de la localidad que queda sin utilizar.*

***Fragmentación Externa (Para el caso de Particiones Dinámicas)***

Son huecos que van quedando en la memoria a medida que los procesos finalizan. Al no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alocar un proceso, pero que no la podamos utilizar.

Solución a fragmentación externa:

La compactación es una técnica que consiste en reorganizar las particiones y procesos en la memoria para eliminar los espacios libres dispersos y liberar un bloque continuo de memoria libre. Esta técnica se utiliza principalmente en sistemas con particiones de tamaño variable, como las particiones dinámicas.

***Paginacion***

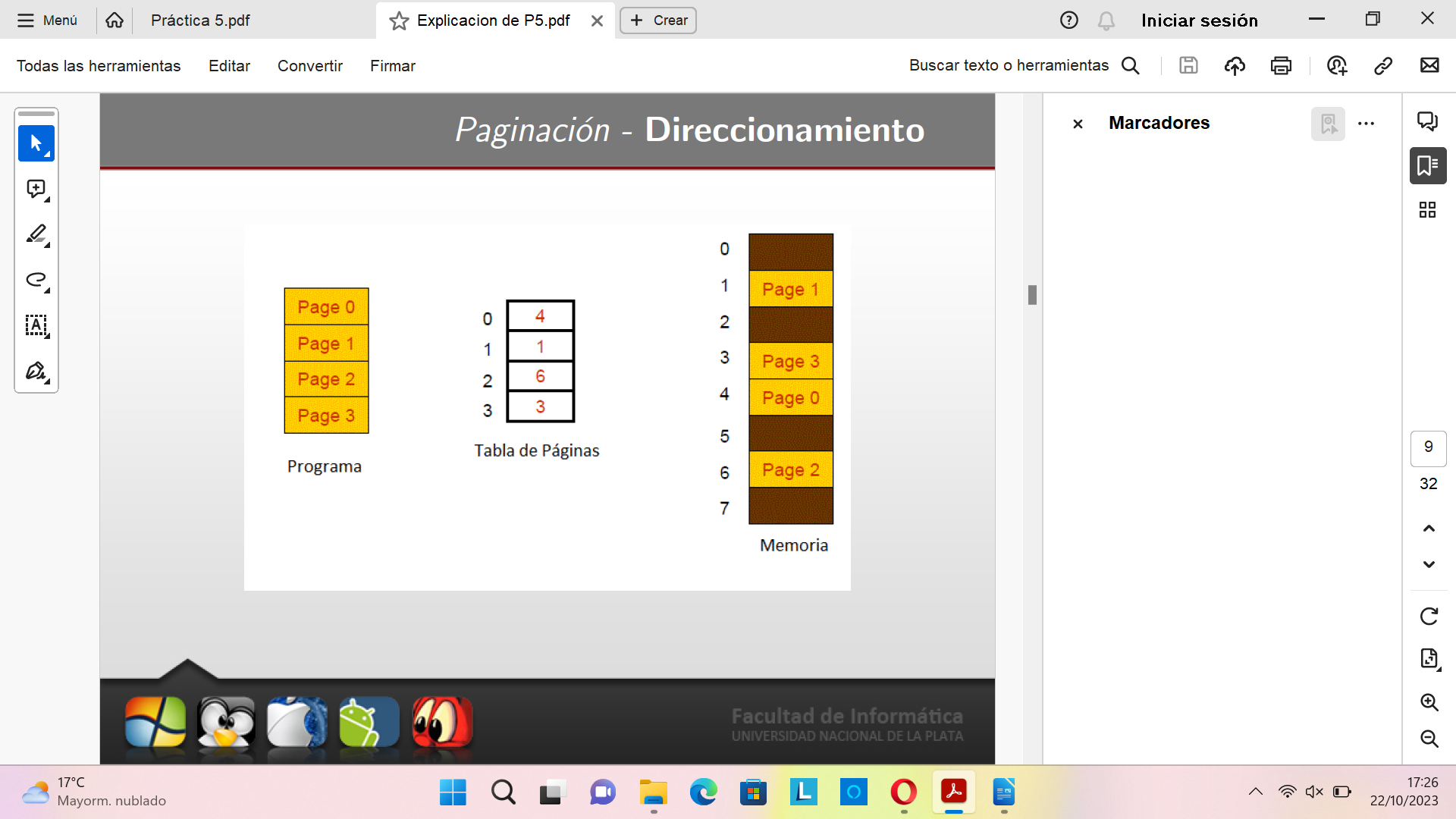
***Explique como trabaja este método de asignación de memoria.***

La memoria se divide en porciones de igual tamaño llamadas marco. El espacio de direcciones de los procesos se divide en porciones de igual tamaño denominadas paginas. Tamaño pagina = tamaño marco = 512 bytes (generalmente).

***Que estructuras adicionales debe poseer el SO para llevar a cabo su implementación?***

El SO mantiene una tabla de paginas para cada proceso, la *cual contiene el marco donde se encuentra cada pagina.*

***Explique, utilizando gráficos, como son transformadas las direcciones lógicas en físicas.***

******

***En este esquema: Se puede producir fragmentación (interna y/o externa)?***

En la paginación, la fragmentación interna generalmente no es un problema, ya que las páginas y marcos de página tienen tamaños fijos y son completamente utilizados. Sin embargo, puede haber fragmentación externa en el almacenamiento secundario, ya que no todos los marcos de página de un proceso pueden residir en memoria principal al mismo tiempo. Esto puede requerir movimientos de páginas entre memoria y almacenamiento secundario, lo que puede generar fragmentación en el almacenamiento secundario.

*Cite similitudes y diferencias entre la técnica de paginacion y la de particiones fijas.*

Similitudes

1. Segmentación de la Memoria: Ambos métodos implican dividir la memoria principal en segmentos o particiones, pero difieren en cómo se realizan estas divisiones.
2. Protección de Memoria: Tanto la paginación como las particiones fijas permiten la protección de la memoria, ya que los procesos no pueden acceder a áreas de memoria asignadas a otros procesos.
3. Asignación de Recursos a Procesos: Ambos métodos se utilizan para asignar espacio de memoria a procesos en ejecución, lo que permite a varios procesos compartir la memoria principal de manera segura.

Diferencias

1. Tamaño de las Divisiones:
   * Paginación: En la paginación, tanto las páginas virtuales como los marcos de página físicos tienen tamaños fijos y uniformes. Las páginas se dividen en tamaño fijo, y la asignación se realiza en función de estas páginas.
   * Particiones Fijas: En las particiones fijas, las divisiones pueden tener tamaños variables, pero son predefinidos y estáticos. Cada partición tiene un tamaño fijo y se asigna a un proceso en su totalidad.
2. Fragmentación:
   * Paginación: Elimina la fragmentación interna ya que se utilizan páginas completas y no hay espacio no utilizado dentro de una página. Sin embargo, puede haber fragmentación externa en el almacenamiento secundario.
   * Particiones Fijas: Puede haber fragmentación interna, ya que los procesos deben ajustarse al tamaño de la partición asignada. Además, puede haber fragmentación externa si las particiones no se utilizan de manera eficiente.
3. Flexibilidad:
   * Paginación: Ofrece más flexibilidad en la asignación de memoria, ya que no es necesario que los procesos se ajusten a particiones de tamaño fijo.
   * Particiones Fijas: Es menos flexible, ya que los procesos deben ajustarse al tamaño de la partición, lo que puede resultar en subutilización o falta de espacio para procesos más grandes.
4. Administración de la Memoria Libre:
   * Paginación: La administración de la memoria libre se realiza en términos de páginas enteras, lo que facilita el seguimiento y la asignación eficiente de memoria.
   * Particiones Fijas: La administración de la memoria libre puede ser más complicada, ya que es necesario llevar un registro detallado de las particiones libres y ocupadas.

***Segmentación***

***Explique como trabaja este método de asignación de memoria.***

La segmentación básicamente la podemos ver como una mejora de la paginacion. Ahora la tabla de segmentos, ademas de tener la dirección de inicio del mismo, tiene la longitud o limite.

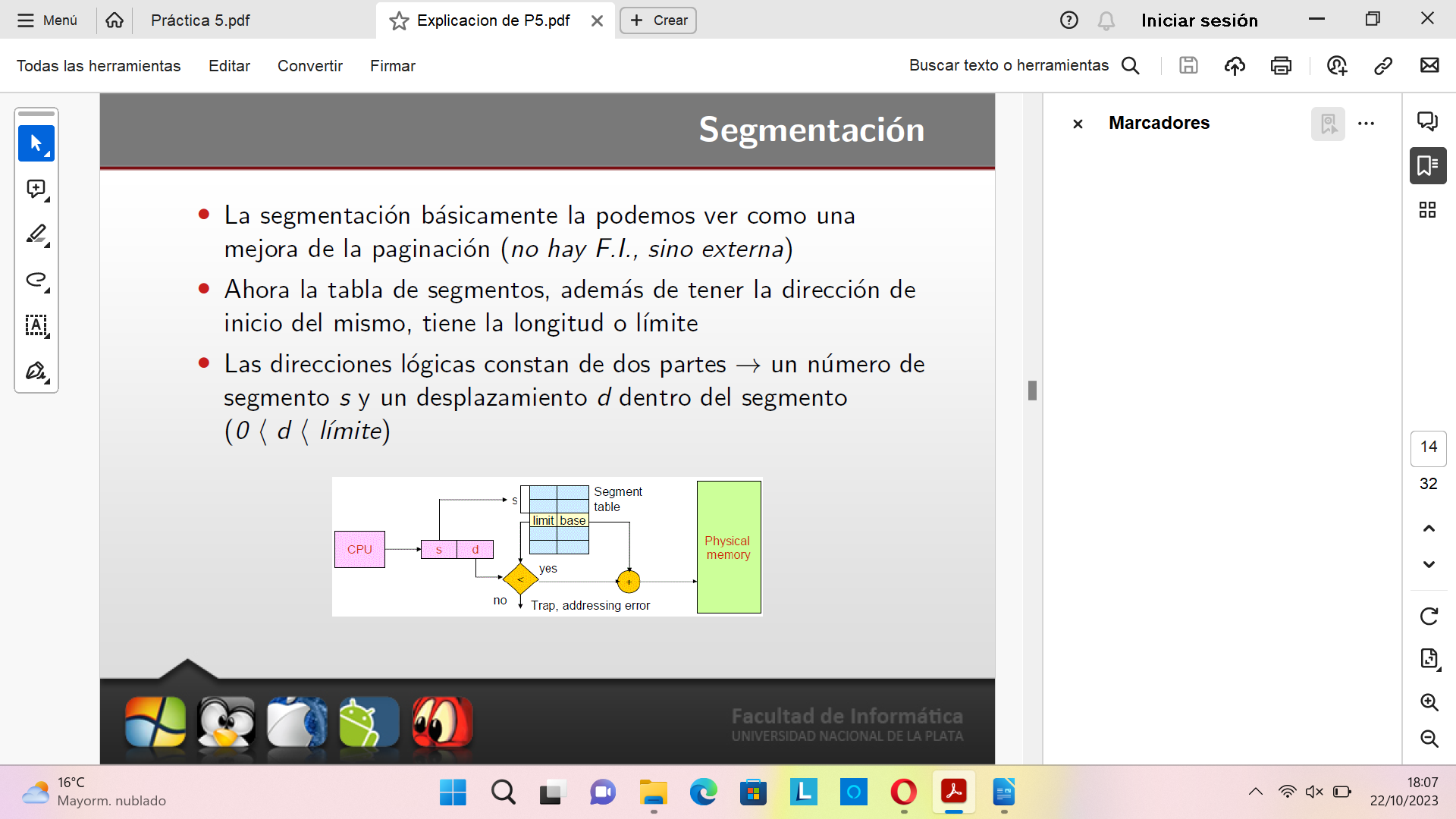
Las direcciones lógicas constan de dos partes → un numero de segmento s y un desplazamiento d dentro del segmento (0 ⟨ d ⟨ limite).

***¿Qué estructuras adicionales debe poseer el SO para llevar a cabo su implementación?***

Estructuras adicionales que debe poseer el sistema operativo para llevar a cabo la implementación de la segmentación:

* Tabla de Segmentos
* Registros de Estado del Proceso
* Gestión de la Protección
* Gestión de Fragmentación

***Explique, utilizando graficos, como son transformadas las direcciones logicas en fisicas.***

******

***En este esquema: ¿Se puede producir fragmentación (interna y/o externa)?***

Fragmentación Interna: En el esquema de segmentación, la fragmentación interna generalmente no es un problema importante. Esto se debe a que los segmentos pueden tener tamaños variables y, en general, se utilizan de manera eficiente. Los segmentos solo tienen el espacio que necesitan, por lo que no se desperdicia memoria.

Fragmentación Externa: En la segmentación, es posible que se produzca fragmentación externa en la memoria principal. Si se asignan muchos segmentos a la memoria y se liberan algunos de ellos a lo largo del tiempo, es posible que se generen espacios vacíos entre los segmentos activos. Esta fragmentación externa puede reducir la eficiencia de la administración de memoria y requerir técnicas como la compactación o la reubicación para abordarla.

En resumen, la segmentación es eficaz para reducir la fragmentación interna, ya que los segmentos se ajustan al tamaño de los datos o instrucciones que contienen. Sin embargo, puede haber fragmentación externa si se asignan y liberan segmentos en la memoria con frecuencia. En ese caso, se pueden utilizar técnicas para mitigar la fragmentación externa y mantener un uso eficiente de la memoria.

***Similitudes y diferencias entre la técnica de segmentación y la de particiones dinámicas:***

Similitudes

Ambas técnicas se utilizan en la gestión de la memoria en sistemas operativos para permitir a múltiples procesos compartir la memoria física de manera eficiente.

Tanto la segmentación como las particiones dinámicas permiten a los programas o procesos acceder a un espacio de direcciones lógicas separado, lo que proporciona aislamiento entre los procesos.

En ambos casos, se maneja la fragmentación, aunque de manera diferente. La segmentación maneja la fragmentación interna, mientras que las particiones dinámicas pueden manejar la fragmentación interna y externa.

Diferencias

La segmentación divide la memoria física en segmentos o regiones lógicas de diferentes tamaños, cada uno asociado a una función especifica del programa (por ejemplo, código, datos, pila).

Cada proceso puede consistir en múltiples segmentos de tamaños variables.

La gestión de la memoria se vuelve más flexible, ya que los segmentos pueden crecer o reducirse según las necesidades del proceso.

La fragmentaciónón interna es una preocupación en la segmentación, ya que cada segmento puede tener su propia fragmentación interna.

Las particiones dinámicas dividen la memoria en regiones de tamaños variables, pero estas regiones no están necesariamente asociadas a funciones específicas del programa.

Cada proceso se carga en una partición de tamaño adecuado que se ajusta a sus necesidades de memoria.

La memoria no se comparte entre procesos, y la gestión se vuelve más rígida en comparación con la segmentación, ya que no se pueden cambiar los tamaños de las particiones después de la asignación inicial.

Las particiones dinámicas pueden sufrir de fragmentación externa, donde hay espacios no utilizables entre particiones.

***Similitudes y diferencias entre la técnica de paginación y segmentación:***

Similitudes

Ambas técnicas se utilizan para gestionar la memoria virtual en sistemas operativos modernos.

Tanto la paginación como la segmentación permiten a los programas o procesos acceder a direcciones lógicas separadas, lo que proporciona aislamiento entre los procesos.

Diferencias

En la paginación, la memoria se divide en páginas de tamaño fijo y se divide en bloques del mismo tamaño en la memoria física.

Las páginas son unidades de asignación y se utilizan para asignar y liberar espacio en la memoria física de manera más eficiente.

La fragmentación interna no es un problema en la paginación, pero puede haber fragmentación externa en el sistema de memoria virtual.

En la segmentación, la memoria se divide en segmentos de diferente tamaño, y cada segmento se asocia con una función lógica del programa.

Los segmentos se asignan en la memoria física, y los segmentos de un proceso pueden variar en tamaño y forma a lo largo de su ejecución.

La fragmentación interna es una preocupación en la segmentación, ya que cada segmento puede tener su propia fragmentación interna.

La segmentación se utiliza para proporcionar una estructura lógica más flexible en comparación con la paginación, que se enfoca en la asignación de memoria en bloques fijos.

***Memoria Virtual***

***Beneficios de la Memoria Virtual***

La memoria virtual es un esquema de administración de la memoria que aporta varios beneficios clave en sistemas operativos. Algunos de los beneficios más destacados incluyen:

* Mayor utilización de la memoria física: La memoria virtual permite que los procesos utilicen más memoria de la que está físicamente disponible. Los programas pueden acceder a una dirección lógica más grande que la memoria física, lo que facilita la ejecución de aplicaciones más grandes y la gestión eficiente de la memoria.
* Aislamiento de procesos: La memoria virtual permite aislar los procesos entre sí. Cada proceso tiene su propio espacio de direcciones lógicas, lo que evita que un proceso dañe accidentalmente la memoria de otro.
* Facilita la gestión de la memoria: Permite una gestión más eficiente de la memoria física al cargar en la memoria solo las partes necesarias de un programa en un momento dado. Esto se traduce en una mejor asignación de recursos y una ejecución más rápida de los procesos.
* Simplifica la administración de la memoria: Los sistemas operativos pueden utilizar técnicas como la paginación o la segmentación para administrar la memoria virtual de manera más eficiente, lo que reduce la complejidad de la gestión de la memoria física.
* Flexibilidad en la ubicación de procesos: Los procesos no necesitan ubicarse en ubicaciones contiguas en la memoria física, lo que permite una carga y descarga más flexible de programas y datos.
* Mayor multitarea: La memoria virtual facilita la ejecución de múltiples procesos en un sistema, lo que mejora la multitarea y la eficiencia de los sistemas operativos.

***Apoyo del Sistema Operativo para la Implementación de la Memoria Virtual***

Para implementar la memoria virtual, el sistema operativo debe proporcionar varios componentes y funcionalidades clave, que incluyen:

* Administración de la memoria física
* Mapeo de direcciones
* Administración de la paginación o segmentación
* Algoritmos de reemplazo de páginas
* Gestión de la memoria virtual
* Protección y aislamiento de procesos
* Gestión de errores de página

***Al implementar esta técnica utilizando paginación por demanda, las tablas de páginas de un proceso deben contar con información adicional además del marco donde se encuentra la página. ¿Cuál es esta información? ¿Porque es necesaria?***

* Bit de presencia (o bit de validez): Este bit indica si la página está actualmente en memoria física (presente) o si ha sido trasladada al almacenamiento secundario (ausente). Es necesario para determinar si una página debe cargarse desde el almacenamiento secundario cuando se accede por primera vez o si ya está en memoria física.
* Bits de protección: Estos bits definen los permisos de acceso a la página, como lectura, escritura o ejecución. Se utilizan para garantizar la seguridad y protección de la memoria de un proceso.
* Bit de modificación (o bit dirty): Este bit indica si la página ha sido modificada desde que se cargó en memoria física. Es necesario para determinar si se debe escribir la página nuevamente en el almacenamiento secundario antes de liberar su marco.
* Bit de referencia (o bit de uso): Este bit se establece cuando se accede a la página. Se utiliza en algoritmos de reemplazo de páginas (como el algoritmo LRU) para determinar cuáles son las páginas menos utilizadas y, por lo tanto, candidatas para ser reemplazadas en caso de falta de espacio en memoria física.

***Fallos de Pagina (Page Faults):***

***¿Cuándo se producen?***

Los fallos de página (page faults) ocurren en sistemas de memoria virtual cuando un programa o proceso intenta acceder a una página de memoria que no se encuentra actualmente en memoria física.

***¿Quién es responsable de detectar un fallo de página?***

La responsabilidad de detectar un fallo de página recae en el hardware del sistema y en el sistema operativo.

***Describa las acciones que emprende el SO cando se produce un fallo de página.***

1. Identificar la causa del fallo de página: El sistema operativo primero debe determinar por qué se produjo el fallo de página. Puede ser el resultado de un intento de acceso a una página que no se encuentra en la memoria física, o puede deberse a otros factores, como un error de programación, una violación de permisos, etc.
2. Verificar la legalidad del acceso: El sistema operativo verifica si el acceso a la página es legítimo y cumple con los permisos de seguridad. Si el acceso no es válido (por ejemplo, un intento de escritura en una página de solo lectura), se genera un error y el proceso se puede suspender o finalizar.
3. Determinar la ubicación de la página: Si el acceso a la página es válido pero la página no está en la memoria física (RAM), el sistema operativo debe determinar la ubicación de la página en el almacenamiento secundario (generalmente un disco duro).
4. Cargar la página desde el almacenamiento secundario: El sistema operativo inicia el proceso de cargar la página requerida desde el almacenamiento secundario a la memoria física. Esto implica la lectura de la página desde el disco y su escritura en un marco de página disponible en la RAM.
5. Actualizar las tablas de páginas: Una vez que la página ha sido cargada en la memoria física, el sistema operativo actualiza las tablas de páginas del proceso para reflejar la nueva ubicación de la página en la memoria física. Esto incluye la modificación del bit de presencia (o validez) en la tabla de páginas para indicar que la página ahora está en memoria.
6. Reanudar la ejecución del proceso: Después de cargar la página y actualizar las tablas de páginas, el sistema operativo permite que el proceso que experimentó el fallo de página continúe su ejecución desde donde se detuvo debido al fallo de página. El proceso ahora puede acceder a la página en memoria física sin problemas.
7. Manejar casos excepcionales: En casos excepcionales, como falta de espacio en memoria física, el sistema operativo puede necesitar seleccionar una página para reemplazar. Para ello, utiliza algoritmos de reemplazo de páginas, como el algoritmo FIFO o LRU, para liberar un marco de página antes de cargar la página requerida.

***Reemplazo de páginas (selección de una víctima)***

¿Qué sucede cuando todos los marcos en la memoria principal están usados por las páginas de los procesos y se produce en fallo de página? El SO debe seleccionar una de las páginas que se encuentra en memoria como víctima, y ser reemplazada por la nueva página que produjo el fallo.

***Clasifique estos algoritmos de malo a bueno de acuerdo a la tasa de fallos de pagina que se obtienen al utilizarlos.***

* FIFO (First-In, First-Out): FIFO se clasificaría como uno de los peores algoritmos en términos de tasa de fallos de página. Esto se debe a que no tiene en cuenta el historial de acceso de las páginas y, por lo tanto, puede reemplazar páginas activas que aún se utilizan con frecuencia.
* LRU (Least Recently Used): LRU tiende a proporcionar mejores resultados en términos de tasa de fallos de página en comparación con FIFO. Esto se debe a que LRU reemplaza la página que no se ha accedido durante más tiempo. Sin embargo, la implementación real de LRU puede ser costosa en cuanto a tiempo y recursos, lo que limita su eficiencia en sistemas grandes.
* Segunda Chance (Clock): Segunda Chance es una variante de FIFO que proporciona un mejor rendimiento que FIFO al considerar si una página ha sido accedida recientemente. Aunque es más efectivo que FIFO, generalmente se clasificaría por debajo de LRU.
* OPT (Optimo): El algoritmo OPT, que reemplaza la página que no se usará durante más tiempo en el futuro, se considera el mejor en términos de tasa de fallos de página teóricos. Sin embargo, su implementación real es imposible en la mayoría de los sistemas, ya que requeriría conocer el futuro patrón de acceso a las páginas.

***Analice su funcionamiento. ¿Como los implementaría?***

* FIFO (First-In, First-Out): FIFO reemplaza la página más antigua en la memoria, como si fuera una cola. La implementación implica mantener un puntero a la página más antigua, y cuando se produce un fallo de página, se reemplaza esa página. Es simple pero no considera el comportamiento de acceso real.
* LRU (Least Recently Used): LRU reemplaza la página que no se ha accedido durante más tiempo. Implementarlo de manera óptima puede ser costoso, ya que requiere un seguimiento completo del historial de acceso a las páginas.
* Segunda Chance (Clock): Este algoritmo es una variante de FIFO que se basa en un bit de "segunda oportunidad". Si la página ha sido accedida recientemente, se le da una "segunda oportunidad" y no se reemplaza. La implementación se basa en el mantenimiento de una cola circular.
* OPT (Optimo): OPT, en teoría, reemplaza la página que no se usará durante más tiempo en el futuro. Sin embargo, su implementación es impracticable, ya que requiere conocimiento completo del futuro comportamiento de acceso.

***Sabemos que la pagina a ser reemplaza puede estar modificada. ¿Qué acciones debe llevar el SO cuando se encuentra ante esta situación?***

* Escribir la página modificada en el almacenamiento secundario: Antes de reemplazar la página, el sistema operativo debe escribir el contenido de la página modificada en el almacenamiento secundario (generalmente en el disco duro) para que los cambios no se pierdan.
* Actualizar las tablas de páginas: Después de escribir la página en el almacenamiento secundario, el sistema operativo debe actualizar las tablas de páginas para reflejar que la página ya no se encuentra en memoria física y que su ubicación está en el almacenamiento secundario.
* Liberar el marco de página en la memoria física: Una vez que se ha realizado la escritura de la página y se han actualizado las tablas de páginas, el marco de página en la memoria física puede liberarse y estar disponible para cargar nuevas páginas.

***Asignación de marcos a un proceso (Conjunto de trabajo o Working Set)***

Con la memoria virtual paginada, no se requiere que todas las páginas de un proceso se encuentren en memoria. El SO debe controlar cuantas páginas de un proceso puede tener en la memoria principal. Existen 2 políticas que se pueden utilizar:

* Asignación Fija: a cada proceso se le asigna una cantidad arbitraria de marco. A su vez para el reparto se puede usar:
  + Reparto equitativo: se asigna la misma cantidad de marcos a cada proceso → m div p
  + Reparto proporcional: se asignan marco en base a la necesidad que tiene cada proceso → Vp . m / Vt
* Asignación Dinámica: los procesos se van cargando en forma dinámica de acuerdo a la cantidad de marcos que necesiten.

***Alcance del reemplazo***

* Reemplazo local: El fallo de página de un proceso solo puede reemplazar sus propias páginas.
* Reemplazo global: el fallo de página de un proceso puede reemplazar la página de cualquier proceso.

***¿Es posible utilizar la política de “Asignación Fija” de marcos junto con la política de “Reemplazo Global?***

Sí, es posible utilizar la política de "Asignación Fija" de marcos junto con la política de "Reemplazo Global" en sistemas de administración de memoria, pero estas dos políticas operan en diferentes aspectos de la gestión de memoria y tienen sus propios propósitos específicos.

***Hiperpaginacion (Trashing)***

Un sistema está en thrashing cuando pasa más tiempo paginando que ejecutando procesos.

***¿Cuáles pueden ser los motivos que la causan?***

Si un proceso no cuenta con todos los frames que necesita, habría thrashing. También con la anomalía de

Belady.

***¿Como la detecta el SO?***

Para detectar la hiperpaginación, el sistema operativo puede utilizar métricas y algoritmos que supervisen el comportamiento de la memoria y las tasas de fallos de página.

***Una vez que lo detecta ¿Que acciones puede tomar el SO para eliminar este problema?***

Las referencias a datos y programas dentro de un proceso tienden a agruparse. La localidad de un proceso en un momento dado se da por el conjunto de páginas que son referenciadas en ese momento. En cortos periodos de tiempo, el proceso necesitara pocas “piezas” del proceso (una p´agina de instrucciones y otra de datos). Entonces se define una ventana de trabajo o working set (Δ) que contiene las referencias de memoria más recientes. Working Set: es el conjunto de p´aginas que tienen las Δ

referencias a páginas más recientes.

***Considere un espacio lógico de 8 páginas de 1024 bytes cada una, mapeadas en una memoria física de 32***

***marcos.***

***¿Cuántos bits son necesarios para representar una dirección lógica?***

Para representar una dirección lógica, necesitas saber cuántas páginas diferentes puedes referenciar en tu espacio lógico. En este caso, tienes 8 páginas. Para representar 8 páginas, necesitarás 3 bits, ya que 2^3 = 8. Entonces, se necesitan 3 bits para representar una dirección lógica.

***¿Cuántos bits son necesarios para representar una dirección física?***

Para representar una dirección física, debes considerar cuántos marcos diferentes puedes referenciar en tu memoria física. Tienes 32 marcos en la memoria física. Para representar 32 marcos, necesitarás 5 bits, ya que 2^5 = 32. Entonces, se necesitan 5 bits para representar una dirección física.

***Direcciones***

***Si se dispone de un espacio de direcciones virtuales de 32 bits, donde cadadirección referencia 1 byte ¿Cuál es el tamaño máximo de un proceso (recordar “espacio virtual”)?***

Si tienes una dirección virtual de 32 bits, puedes representar 2^32 direcciones diferentes. Cada dirección referencia 1 byte. Entonces, el tamaño máximo de un proceso sería de: Tamaño máximo = 2^32 bytes = 4 gigabytes (GB).

***Si el tamaño de página es de 512Kb. ¿Cuál es el número máximo de páginas que puede tener un proceso?***

Si el tamaño de página es de 512 kilobytes (512 KB), primero debemos convertirlo a bytes: Tamaño de página = 512 KB = 512 \* 1024 bytes = 524,288 bytes. El número máximo de páginas se puede calcular dividiendo el espacio virtual por el tamaño de página:

Número máximo de páginas = Tamaño máximo / Tamaño de página Número máximo de páginas = (2^32 bytes) / (524,288 bytes/página) ≈ 8,192 páginas.

***Si el tamaño de página es de 512Kb. y se disponen de 256 Mb. De memoria real. ¿Cuál es el número de marcos que puede haber?***

Si tienes 256 megabytes (256 MB) de memoria real, primero debemos convertirlo a bytes: Memoria real = 256 MB = 256 \* 1024 \* 1024 bytes = 268,435,456 bytes.

El número de marcos que puedes tener se calcula dividiendo la memoria real por el tamaño de página: Número de marcos = Memoria real / Tamaño de página Número de marcos = (268,435,456 bytes) / (524,288 bytes/marco) ≈ 512 marcos.

***Si se utilizaran 2 Kb. para cada entrada en la tabla de páginas de un proceso: ¿Cuál sería el tamaño máximo de la tabla de páginas de cada proceso?***

Si se utilizan 2 kilobytes (2 KB) para cada entrada en la tabla de páginas, entonces el tamaño máximo de la tabla de páginas de cada proceso se calcula dividiendo el tamaño máximo del proceso entre el tamaño de cada entrada en la tabla de páginas:

Tamaño máximo de la tabla de páginas = Tamaño máximo del proceso / Tamaño de entrada en la tabla de páginas Tamaño máximo de la tabla de páginas = (4 GB) / (2 KB/entrada) ≈ 2 megabytes (MB).

Entonces, el tamaño máximo de la tabla de páginas de cada proceso sería de aproximadamente 2 megabytes.