Sistemas Operativos I

Gestión de memoria

Edwin Salvador

17 de diciembre de 2015

Sesión 12

Contenido I

Administrador de memoria

Figura: Administrador de memoria

Aumento de memoria

- Los programas se expanden para llenar la memoria disponible para contenerlos.
- Mientras más memoria tenemos los programas requieren más memoria para ejecutarse en lugar de poder ejecutar más programas.
- Antes un programa de usuario podria necesitar entre 5 y 10 MB de memoria, en la actualidad pueden necesitar desde 50 a 200 MB o más
- El administrador de memoria debe manejarla de manera eficiente.
  - llevar el registro de cuáles partes de la memoria están en uso,
  - asignar memoria a los procesos cuando la necesiten y
  - desasignarla cuando terminen.

Contenido I

Requisitos de la gestión de la memoria

- Los requisitos que la gestión de memoria debe satisfacer son:
  - Reubicación
  - Protección
  - Compartición
  - Organización lógica
  - Organización física

Contenido I

Reubicación

- En los sistemas multiprogramados la memoria principal se comparte entre varios procesos. Para un programador no es posible saber con exactitud que programas residirán en memoria principal al momento de ejecutar su programa.

Reubicación

- En los sistemas multiprogramados la memoria principal se comparte entre varios procesos. Para un programador no es posible saber con exactitud que programas residirán en memoria principal al momento de ejecutar su programa.
- Debido al **intercambio** o *swap*, los programas estarán siendo llevados de la memoria principal al disco, por lo tanto se debe permitir que los programas se puedan mover en la memoria principal. Es decir, se debe poder reubicar los procesos en áreas de memoria diferentes.

Reubicación

- En los sistemas multiprogramados la memoria principal se comparte entre varios procesos. Para un programador no es posible saber con exactitud que programas residirán en memoria principal al momento de ejecutar su programa.
- Debido al **intercambio** o *swap*, los programas estarán siendo llevados de la memoria principal al disco, por lo tanto se debe permitir que los programas se puedan mover en la memoria principal. Es decir, se debe poder reubicar los procesos en áreas de memoria diferentes.
- Esto quiere decir que el SO deberá conocer la ubicación la información de control del proceso, de la pila de ejecución y el punto de entrada que utilizará el proceso para iniciar la ejecución.

Reubicación

- En los sistemas multiprogramados la memoria principal se comparte entre varios procesos. Para un programador no es posible saber con exactitud que programas residirán en memoria principal al momento de ejecutar su programa.
- Debido al **intercambio** o *swap*, los programas estarán siendo llevados de la memoria principal al disco, por lo tanto se debe permitir que los programas se puedan mover en la memoria principal. Es decir, se debe poder reubicar los procesos en áreas de memoria diferentes.
- Esto quiere decir que el SO deberá conocer la ubicación la información de control del proceso, de la pila de ejecución y el punto de entrada que utilizará el proceso para iniciar la ejecución.
- Adicionalmente, el procesador trata con referencias de memoria dentro del programa (instrucciones a ejecutar, datos). El procesador y el SO deben traducir estas referencias en e código a direcciones de memoria físicas las cuales reflejan la ubicación actual del programa.

Contenido I

Protección

- Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos. Los programas de otros procesos no deben ser capaces de referenciar sin permiso posiciones de memoria de un proceso, tanto en modo lectura como escritura.

Protección

- Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos. Los programas de otros procesos no deben ser capaces de referenciar sin permiso posiciones de memoria de un proceso, tanto en modo lectura como escritura.
- El requisito de reubicación hace más complicado cumplir con el requisito de protección.

Protección

- Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos. Los programas de otros procesos no deben ser capaces de referenciar sin permiso posiciones de memoria de un proceso, tanto en modo lectura como escritura.
- El requisito de reubicación hace más complicado cumplir con el requisito de protección.
- Todas las referencias de memoria generadas por un proceso deben comprobarse en tiempo de ejecución para asegurarse que se refieren solo la espacio de memoria asignado a ese proceso.

Protección

- Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos. Los programas de otros procesos no deben ser capaces de referenciar sin permiso posiciones de memoria de un proceso, tanto en modo lectura como escritura.
- El requisito de reubicación hace más complicado cumplir con el requisito de protección.
- Todas las referencias de memoria generadas por un proceso deben comprobarse en tiempo de ejecución para asegurarse que se refieren solo la espacio de memoria asignado a ese proceso.
- Un programa de un proceso no puede saltar a una instrucción de otro proceso ni a los datos de otro proceso (sin un trato especial). El procesador deber ser capaz de abortar este tipo de instrucciones en el pinto de ejecución.

Protección

- Los requisitos de protección deben ser satisfechos por el procesador (hardware) y no por el SO (software). Esto es porqué tomaría mucho tiempo al SO anticipar todas las referencias de memoria que un programa hará para detectar si existen violaciones.
- Solo es posible evaluar la validez de una referencia (acceso a datos o salto de instrucciones) en tiempo de ejecución de la instrucción que realiza la referencia.
- Los mecanismos para satisfacer la reubicación también dan soporte al requisito de protección.

Contenido I

Compartición

- Un mecanismo de protección debe tener la flexibilidad de permitir a varios procesos acceder a la misma porción de memoria principal.
- Por ejemplo, si varios procesos están ejecutando el mismo programa, todos deberían tener acceso a la misma copia del programa en lugar de tener una copia para cada uno.
- El sistema de gestión de la memoria debe permitir el acceso controlado a áreas de memoria compartidas sin comprometer la protección esencial.
- Los mecanismos para la reubicación también dan soporte a la campartición.

Contenido I

Organización lógica

- Contrario a la organización de la memoria principal, la mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que se pueden modificar.

Organización lógica

- Contrario a la organización de la memoria principal, la mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que se pueden modificar.
- Si el sistema operativo y el hardware del computador pueden tratar de forma efectiva los programas de usuarios y los datos en la forma de módulos, se pueden lograr 3 ventajas:

Organización lógica

- Contrario a la organización de la memoria principal, la mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que se pueden modificar.
- Si el sistema operativo y el hardware del computador pueden tratar de forma efectiva los programas de usuarios y los datos en la forma de módulos, se pueden lograr 3 ventajas:
  1. Los módulos se pueden escribir y compilar independientemente, con todas las referencias de un módulo desde otro resueltas por el sistema en tiempo de ejecución.

Organización lógica

- Contrario a la organización de la memoria principal, la mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que se pueden modificar.
- Si el sistema operativo y el hardware del computador pueden tratar de forma efectiva los programas de usuarios y los datos en la forma de módulos, se pueden lograr 3 ventajas:
  1. Los módulos se pueden escribir y compilar independientemente, con todas las referencias de un módulo desde otro resueltas por el sistema en tiempo de ejecución.
  2. Con una ligera sobrecarga, se puede proporcionar diferentes grados de protección a los módulos (sólo lectura, sólo ejecución).

Organización lógica

- Contrario a la organización de la memoria principal, la mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que se pueden modificar.
- Si el sistema operativo y el hardware del computador pueden tratar de forma efectiva los programas de usuarios y los datos en la forma de módulos, se pueden lograr 3 ventajas:
  1. Los módulos se pueden escribir y compilar independientemente, con todas las referencias de un módulo desde otro resueltas por el sistema en tiempo de ejecución.
  2. Con una ligera sobrecarga, se puede proporcionar diferentes grados de protección a los módulos (sólo lectura, sólo ejecución).
  3. Es posible introducir mecanismos por los cuales los módulos se pueden compartir entre los procesos.

Organización lógica

- Contrario a la organización de la memoria principal, la mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que se pueden modificar.
- Si el sistema operativo y el hardware del computador pueden tratar de forma efectiva los programas de usuarios y los datos en la forma de módulos, se pueden lograr 3 ventajas:
  1. Los módulos se pueden escribir y compilar independientemente, con todas las referencias de un módulo desde otro resueltas por el sistema en tiempo de ejecución.
  2. Con una ligera sobrecarga, se puede proporcionar diferentes grados de protección a los módulos (sólo lectura, sólo ejecución).
  3. Es posible introducir mecanismos por los cuales los módulos se pueden compartir entre los procesos.
- La herramienta que más adecuadamente satisface estos requisitos es la segmentación, que veremos más adelante.

Contenido I

Organización física

- La memoria del computador se organiza en al menos dos niveles (principal y secundaria). La principal contiene datos actualmente en uso y la secundaria almacena programas y datos a largo plazo.

Organización física

- La memoria del computador se organiza en al menos dos niveles (principal y secundaria). La principal contiene datos actualmente en uso y la secundaria almacena programas y datos a largo plazo.
- Este esquema hace que el flujo entre memoria principal y secundaria sea una de la principales preocupaciones del sistema. Se podría asignar la responsabilidad a cada programador pero esto no es deseable por dos motivos:

Organización física

- La memoria del computador se organiza en al menos dos niveles (principal y secundaria). La principal contiene datos actualmente en uso y la secundaria almacena programas y datos a largo plazo.
- Este esquema hace que el flujo entre memoria principal y secundaria sea una de la principales preocupaciones del sistema. Se podría asignar la responsabilidad a cada programador pero esto no es deseable por dos motivos:
  1. La memoria principal para un programa podría ser insuficiente. Para superar esto, los programadores tendrían que utilizar complicadas técnicas (overlaying) que malgastaría el tiempo del programador.

Organización física

- La memoria del computador se organiza en al menos dos niveles (principal y secundaria). La principal contiene datos actualmente en uso y la secundaria almacena programas y datos a largo plazo.
- Este esquema hace que el flujo entre memoria principal y secundaria sea una de la principales preocupaciones del sistema. Se podría asignar la responsabilidad a cada programador pero esto no es deseable por dos motivos:
  1. La memoria principal para un programa podría ser insuficiente. Para superar esto, los programadores tendrían que utilizar complicadas técnicas (overlaying) que malgastaría el tiempo del programador.
  2. En un entorno multiprogramado, el programador no conoce en tiempo de codificación cuánto espacio estará disponible o dónde se localizará dicho espacio.

Organización física

- La memoria del computador se organiza en al menos dos niveles (principal y secundaria). La principal contiene datos actualmente en uso y la secundaria almacena programas y datos a largo plazo.
- Este esquema hace que el flujo entre memoria principal y secundaria sea una de la principales preocupaciones del sistema. Se podría asignar la responsabilidad a cada programador pero esto no es deseable por dos motivos:
  1. La memoria principal para un programa podría ser insuficiente. Para superar esto, los programadores tendrían que utilizar complicadas técnicas (overlaying) que malgastaría el tiempo del programador.
  2. En un entorno multiprogramado, el programador no conoce en tiempo de codificación cuánto espacio estará disponible o dónde se localizará dicho espacio.
- Por este motivo el flujo de información entre los dos niveles es responsabilidad del sistema de gestión de la memoria.

Contenido I

Sin abstracción de memoria

- Sin una abstracción de memoria, cada programa puede ver simplemente la memoria física.
- No se pueden tener dos programas en memoria ejecutándose simultáneamente.
- Cada programa sobreescribe al actual en memoria para poder ser ejecutado.
- Se puede utilizar un mecanismo como el swapping para simular la ejecución concurrente de programas.

Espacio de direcciones (abstracción de memoria)

- Para permitir varios programas a la vez, se necesita cumplir con los requisitos de reubicación y protección.
- Para esto existe la opción de abstracción de memoria: espacio de direcciones.
- El espacio de direcciones es el conjunto de direcciones que puede utilizar un proceso para direccionar la memoria.
- Cada proceso tiene su propio espacio de direcciones, independiente de los que pertenecen a otros procesos
- Se puede ver como los números de telefono en distintas ciudades, cada ciudad tiene su propio espacio de direcciones,.
- Así la dirección 28 de un proceso es distinta a la dirección 28 de otro.

Sobrecarga de memoria

- Al no ser posible tener todos los procesos en memoria, se emplean mecanismos para lidiar con la sobrecarga de memoria, uno de ellos es???

Sobrecarga de memoria

- Al no ser posible tener todos los procesos en memoria, se emplean mecanismos para lidiar con la sobrecarga de memoria, uno de ellos es??? el intercambio o swapping

Sobrecarga de memoria

- Al no ser posible tener todos los procesos en memoria, se emplean mecanismos para lidiar con la sobrecarga de memoria, uno de ellos es??? el intercambio o swapping
- Otro mecanismo es la **memoria virtual** similar al swapping, pero perimite ejecutar los procesos aunque no estén cargados por completo en memoria principal.

El intercambio

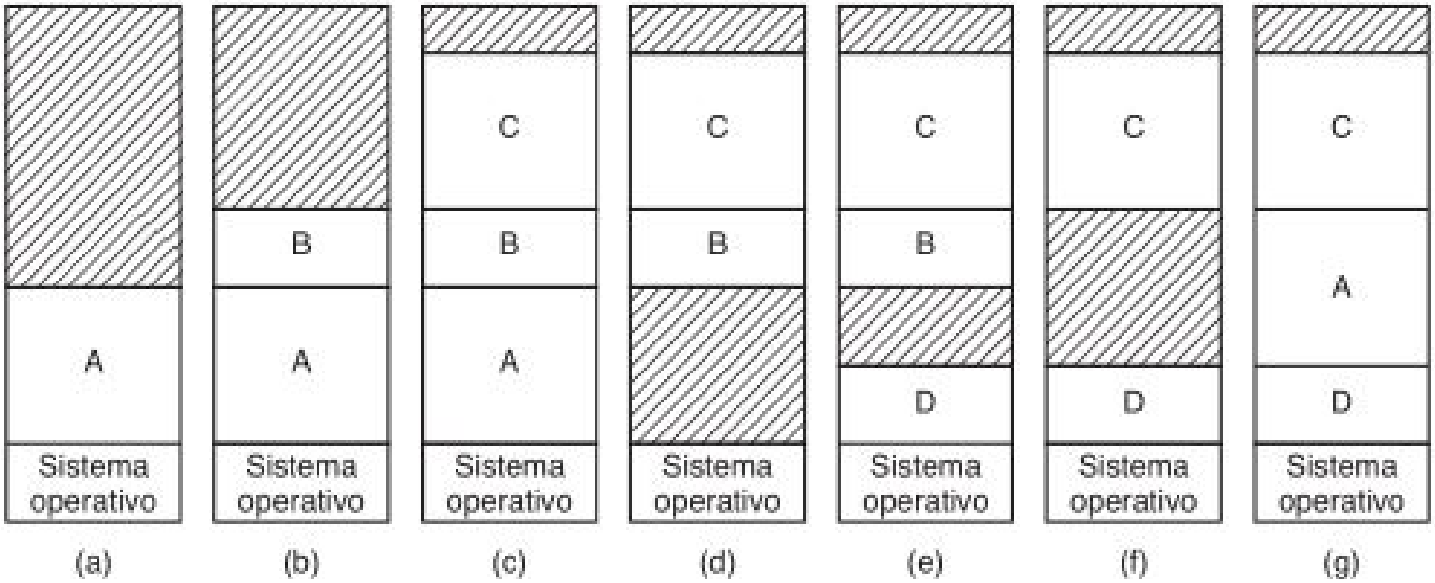


Figura 3-4. La asignación de la memoria cambia a medida que llegan procesos a la memoria y salen de ésta. Las regiones sombreadas son la memoria sin usar.

Compactación de memoria el intercambio crea varios huecos en la memoria, es posible combinarlos todos en uno grande. No se realiza debido con frecuencia, ya que requiere mucho tiempo de la CPU. Una máquina con 1 GB que pueda copiar 4 bytes en 20 nseg, se requerirían aproximadamente 5 segundos para compactar toda la memoria.

Contenido I

Particionamiento

- La principal operación de la gestión de memoria es traer los procesos a la memoria principal para ser ejecutados por el procesador.
- En sistemas multiprogramados esto implica la utilización de la memoria virtual la cuál se basa en técnicas como la segmentación y la paginación que veremos más adelante.
- Antes de estudiar la paginación y segmentación veremos una técnica más sencilla que no utiliza memoria virtual, el particionamiento.

Contenido I

Particionamiento fijo

- El esquema más simple para repartir la memoria principal es repartirla en regiones de tamaños fijos.

Particionamiento fijo

- El esquema más simple para repartir la memoria principal es repartirla en regiones de tamaños fijos.
- Este esquema asume que el SO ocupa una porción fija de la memoria principal y el resto de memoria puede ser utilizada para cargar los procesos.

Particionamiento fijo

- El esquema más simple para repartir la memoria principal es repartirla en regiones de tamaños fijos.
- Este esquema asume que el SO ocupa una porción fija de la memoria principal y el resto de memoria puede ser utilizada para cargar los procesos.
- Existen dos opciones para el particionamiento fijo:

Particionamiento fijo

- El esquema más simple para repartir la memoria principal es repartirla en regiones de tamaños fijos.
- Este esquema asume que el SO ocupa una porción fija de la memoria principal y el resto de memoria puede ser utilizada para cargar los procesos.
- Existen dos opciones para el particionamiento fijo:
  - Particiones del igual tamaño

Particionamiento fijo

- El esquema más simple para repartir la memoria principal es repartirla en regiones de tamaños fijos.
- Este esquema asume que el SO ocupa una porción fija de la memoria principal y el resto de memoria puede ser utilizada para cargar los procesos.
- Existen dos opciones para el particionamiento fijo:
  - Particiones del igual tamaño
  - Particiones de distinto tamaño

Particiones fijas del mismo tamaño

- En este caso, cualquier proceso cuyo tamaño es menor o igual que el tamaño de partición puede cargarse en cualquier partición disponible.

Particiones fijas del mismo tamaño

- En este caso, cualquier proceso cuyo tamaño es menor o igual que el tamaño de partición puede cargarse en cualquier partición disponible.
- Si todas las particiones están llenas y no hay ningún proceso en estado Listo o Ejecutando, el sistema operativo puede mandar a swap a un proceso de cualquiera de las particiones y cargar otro proceso, de forma que el procesador tenga trabajo que realizar.

Particiones fijas del mismo tamaño

- En este caso, cualquier proceso cuyo tamaño es menor o igual que el tamaño de partición puede cargarse en cualquier partición disponible.
- Si todas las particiones están llenas y no hay ningún proceso en estado Listo o Ejecutando, el sistema operativo puede mandar a swap a un proceso de cualquiera de las particiones y cargar otro proceso, de forma que el procesador tenga trabajo que realizar.
- Esta opción tiene dos problemas:

Particiones fijas del mismo tamaño

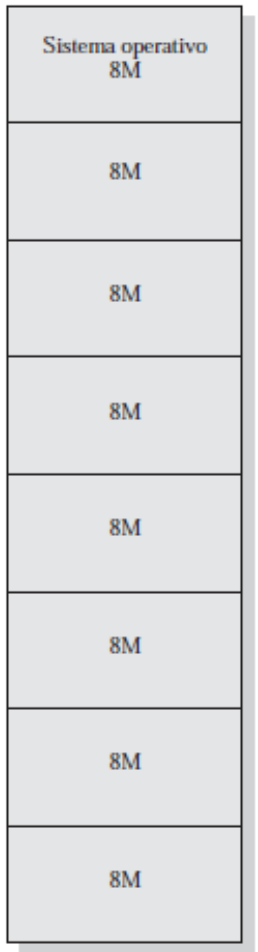
- En este caso, cualquier proceso cuyo tamaño es menor o igual que el tamaño de partición puede cargarse en cualquier partición disponible.
- Si todas las particiones están llenas y no hay ningún proceso en estado Listo o Ejecutando, el sistema operativo puede mandar a swap a un proceso de cualquiera de las particiones y cargar otro proceso, de forma que el procesador tenga trabajo que realizar.
- Esta opción tiene dos problemas:
  - Un programa puede ser más grande que las particiones. En este caso el programador debe diseñar el programa para que pueda ser cargado un módulo a la vez y en ciertas ocasiones necesitará utilizar *overlays* para superponer un módulo o datos en una partición del programa.

Particiones fijas del mismo tamaño

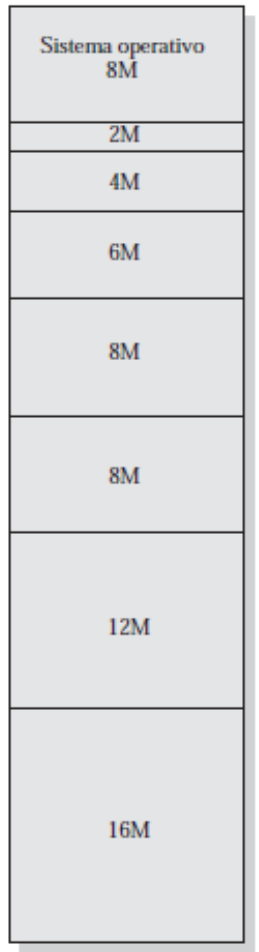
- En este caso, cualquier proceso cuyo tamaño es menor o igual que el tamaño de partición puede cargarse en cualquier partición disponible.
- Si todas las particiones están llenas y no hay ningún proceso en estado Listo o Ejecutando, el sistema operativo puede mandar a swap a un proceso de cualquiera de las particiones y cargar otro proceso, de forma que el procesador tenga trabajo que realizar.
- Esta opción tiene dos problemas:
  - Un programa puede ser más grande que las particiones. En este caso el programador debe diseñar el programa para que pueda ser cargado un módulo a la vez y en ciertas ocasiones necesitará utilizar *overlays* para superponer un módulo o datos en una partición del programa.
  - Se utiliza la memoria principal de manera ineficiente. Los tamaños son fijos por lo tanto no importa si un programa es pequeño, igual tendrá que utilizar la partición entera. A este problema se le conoce como **fragmentación interna**: cuando existe un espacio de memoria malgastado porque contiene un bloque de menor tamaño que la partición.

Particiones de distinto tamaño

- Los dos problemas anteriores pueden mejorarse (**no resolverse**) si se utiliza particiones de tamaño diferente. Así se puede reducir la fragmentación interna.



(a) Participaciones de igual tamaño



(b) Participaciones de distinto tamaño

Algoritmo de ubicación de procesos

- En el caso de tamaño único la ubicación de procesos es simple, se pueden ubicar los procesos en cualquier partición y se puede enviar cualquier proceso a disco si es necesario.

Algoritmo de ubicación de procesos

- En el caso de tamaño único la ubicación de procesos es simple, se pueden ubicar los procesos en cualquier partición y se puede enviar cualquier proceso a disco si es necesario.
- En el caso de tamaño distinto existen dos opciones para la ubicación de procesos:

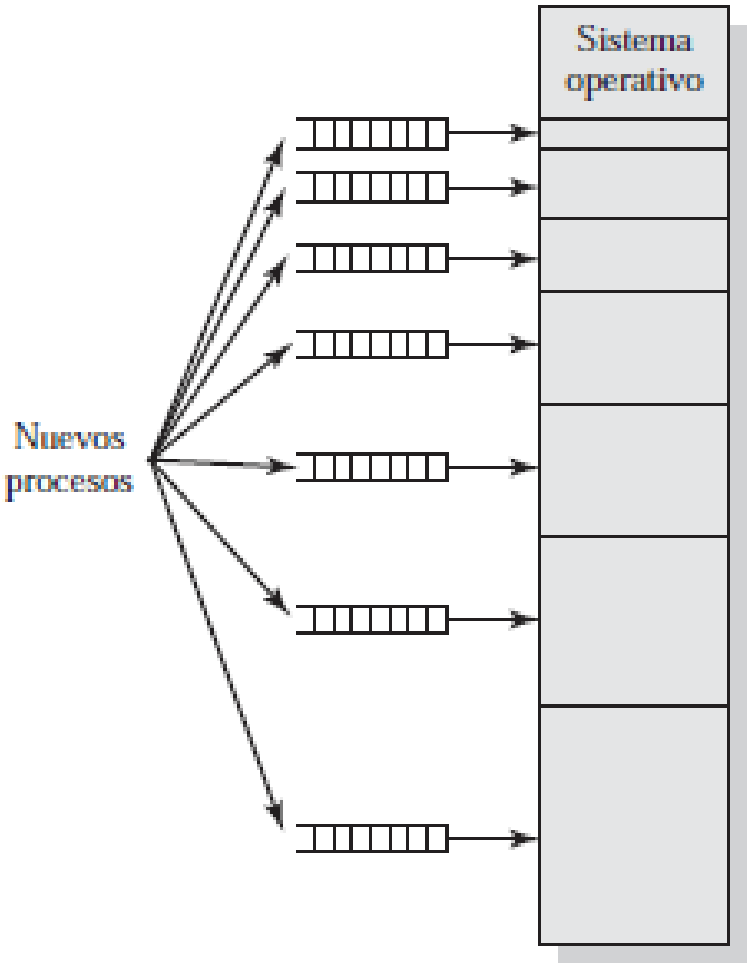
Algoritmo de ubicación de procesos

- En el caso de tamaño único la ubicación de procesos es simple, se pueden ubicar los procesos en cualquier partición y se puede enviar cualquier proceso a disco si es necesario.
- En el caso de tamaño distinto existen dos opciones para la ubicación de procesos:
  - **Crear una cola por cada partición (tamaño)** Se asigna un proceso a la partición más pequeña que lo pueda albergar. Esta es la opción menos óptima ya que pueden quedarse particiones sin utilizar si no existen procesos de ese tamaño.

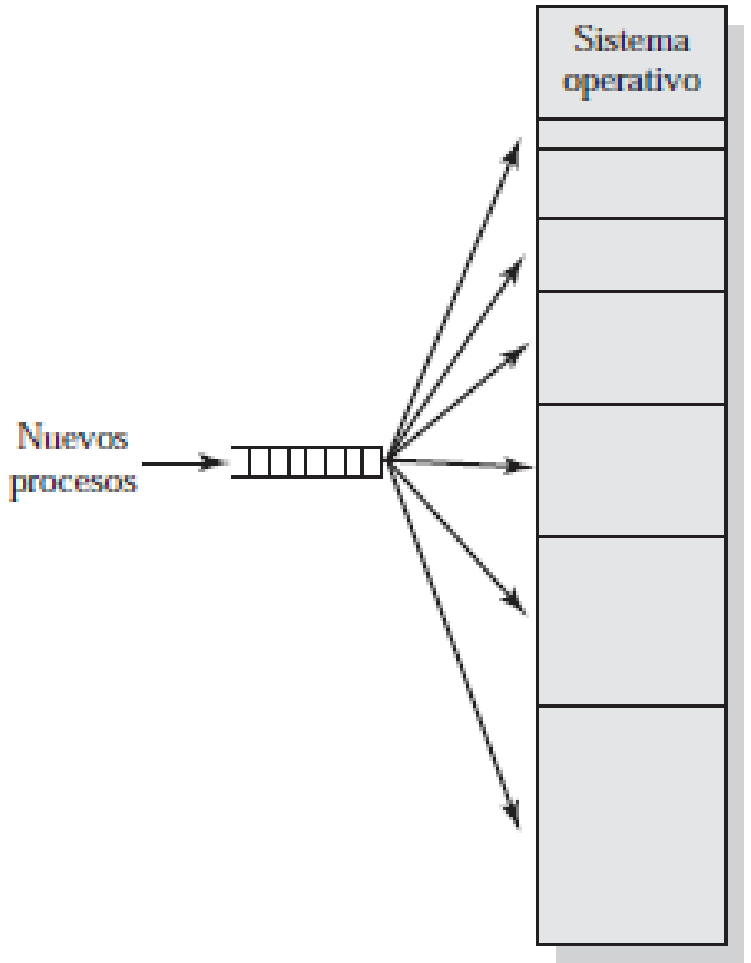
Algoritmo de ubicación de procesos

- En el caso de tamaño único la ubicación de procesos es simple, se pueden ubicar los procesos en cualquier partición y se puede enviar cualquier proceso a disco si es necesario.
- En el caso de tamaño distinto existen dos opciones para la ubicación de procesos:
  - **Crear una cola por cada partición (tamaño)** Se asigna un proceso a la partición más pequeña que lo pueda albergar. Esta es la opción menos óptima ya que pueden quedarse particiones sin utilizar si no existen procesos de ese tamaño.
  - **Crear una única cola para todas las particiones** En el momento de cargar un proceso se selecciona la partición **más pequeña disponible**. Se enviar a swap de preferencia el proceso que ocupe la partición más pequeña que pueda albergar al proceso entrante (se puede considerar también prioridades, procesos bloqueados, etc).

Opciones para ubicación de procesos en particiones



(a) Una cola de procesos por participación



(b) Un única cola

Contenido I

Particionamiento dinámico

- Aunque esta técnica mejora al particionamiento fijo, ya ha sido reemplaza por técnicas de gestión de memoria más sofisticadas.

Particionamiento dinámico

- Aunque esta técnica mejora al particionamiento fijo, ya ha sido reemplaza por técnicas de gestión de memoria más sofisticadas.
- Cuando se carga un proceso se le asigna exactamente tanta memoria como requiera, por lo tanto, las particiones son de longitud y número variable.

Particionamiento dinámico

- Aunque esta técnica mejora al particionamiento fijo, ya ha sido reemplaza por técnicas de gestión de memoria más sofisticadas.
- Cuando se carga un proceso se le asigna exactamente tanta memoria como requiera, por lo tanto, las particiones son de longitud y número variable.
- El sistema empezará repartiendo secciones contiguas de memoria a los procesos hasta que se llegue al punto donde no existirá suficiente espacio para otro proceso y se tendrá que esperar a que se pueda llevar un proceso a disco para asignar este espacio a otro proceso.

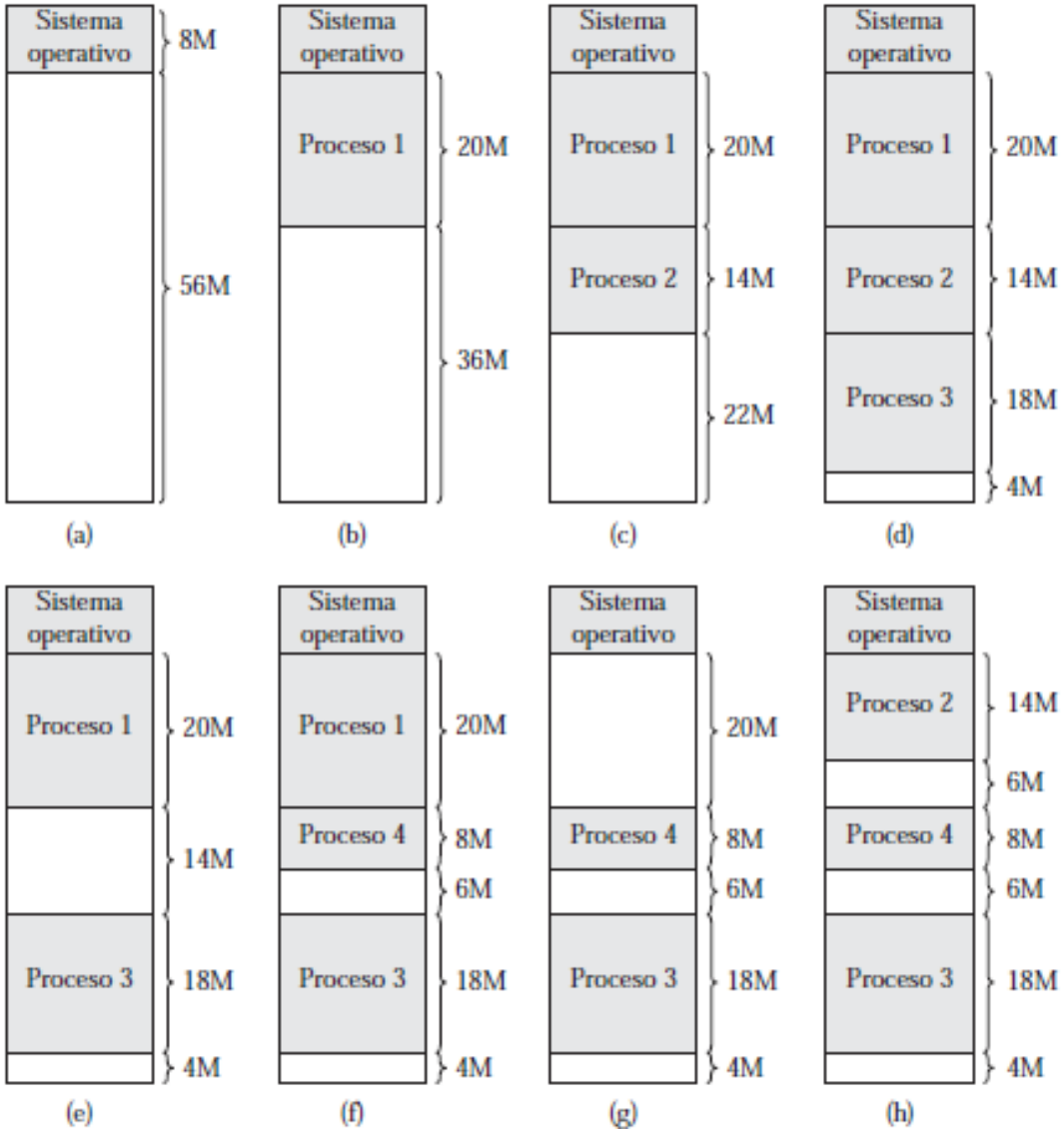
Particionamiento dinámico

- Aunque esta técnica mejora al particionamiento fijo, ya ha sido reemplaza por técnicas de gestión de memoria más sofisticadas.
- Cuando se carga un proceso se le asigna exactamente tanta memoria como requiera, por lo tanto, las particiones son de longitud y número variable.
- El sistema empezará repartiendo secciones contiguas de memoria a los procesos hasta que se llegue al punto donde no existirá suficiente espacio para otro proceso y se tendrá que esperar a que se pueda llevar un proceso a disco para asignar este espacio a otro proceso.
- Esto podrá generar huecos en la memoria muy pequeños que son inutilizables para otros procesos. A medida que pasa el tiempo la memoria se irá fragmentando cada vez más. a este problema se lo conoce como **fragmentación externa** (la memoria externa a todas las particiones se fragmenta de manera incremental).

Particionamiento dinámico

- Aunque esta técnica mejora al particionamiento fijo, ya ha sido reemplaza por técnicas de gestión de memoria más sofisticadas.
- Cuando se carga un proceso se le asigna exactamente tanta memoria como requiera, por lo tanto, las particiones son de longitud y número variable.
- El sistema empezará repartiendo secciones contiguas de memoria a los procesos hasta que se llegue al punto donde no existirá suficiente espacio para otro proceso y se tendrá que esperar a que se pueda llevar un proceso a disco para asignar este espacio a otro proceso.
- Esto podrá generar huecos en la memoria muy pequeños que son inutilizables para otros procesos. A medida que pasa el tiempo la memoria se irá fragmentando cada vez más. a este problema se lo conoce como **fragmentación externa** (la memoria externa a todas las particiones se fragmenta de manera incremental).
- A continuación, la figura muestra como funciona esta técnica.

Particionamiento dinámico



Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.

Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.
- El problema es que la compactación requiere tiempo de procesador para reubicar los procesos. También se debe satisfacer el requerimiento de reubicación dinámica de tal manera que no se pierdan las referencias a memoria de cada programa.

Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.
- El problema es que la compactación requiere tiempo de procesador para reubicar los procesos. También se debe satisfacer el requerimiento de reubicación dinámica de tal manera que no se pierdan las referencias a memoria de cada programa.
- Para decidir en que espacio de memoria ubicar un proceso existen tres opciones:

Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.
- El problema es que la compactación requiere tiempo de procesador para reubicar los procesos. También se debe satisfacer el requerimiento de reubicación dinámica de tal manera que no se pierdan las referencias a memoria de cada programa.
- Para decidir en que espacio de memoria ubicar un proceso existen tres opciones:
  - **Mejor ajuste (*best-fit*)** el bloque más cercano en tamaño.

Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.
- El problema es que la compactación requiere tiempo de procesador para reubicar los procesos. También se debe satisfacer el requerimiento de reubicación dinámica de tal manera que no se pierdan las referencias a memoria de cada programa.
- Para decidir en que espacio de memoria ubicar un proceso existen tres opciones:
  - **Mejor ajuste (*best-fit*)** el bloque más cercano en tamaño.
  - **Primer ajuste (*first-fit*)** Empieza desde el principio y escoge el primer bloque disponible que sea suficientemente grande.

Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.
- El problema es que la compactación requiere tiempo de procesador para reubicar los procesos. También se debe satisfacer el requerimiento de reubicación dinámica de tal manera que no se pierdan las referencias a memoria de cada programa.
- Para decidir en que espacio de memoria ubicar un proceso existen tres opciones:
  - **Mejor ajuste (*best-fit*)** el bloque más cercano en tamaño.
  - **Primer ajuste (*first-fit*)** Empieza desde el principio y escoge el primer bloque disponible que sea suficientemente grande.
  - **Siguiente ajuste (*next-fit*)** comienza a analizar la memoria desde la última colocación y elige el siguiente bloque suficientemente grande.

Particionamiento dinámico

- La **compactación** es una técnica para eliminar la fragmentación externa. El SO desplaza todos los procesos en memoria de manera que todo el espacio libre quede unido en un solo bloque contiguo.
- El problema es que la compactación requiere tiempo de procesador para reubicar los procesos. También se debe satisfacer el requerimiento de reubicación dinámica de tal manera que no se pierdan las referencias a memoria de cada programa.
- Para decidir en que espacio de memoria ubicar un proceso existen tres opciones:
  - **Mejor ajuste (*best-fit*)** el bloque más cercano en tamaño.
  - **Primer ajuste (*first-fit*)** Empieza desde el principio y escoge el primer bloque disponible que sea suficientemente grande.
  - **Siguiente ajuste (*next-fit*)** comienza a analizar la memoria desde la última colocación y elige el siguiente bloque suficientemente grande.
- Cuál de estas técnicas es mejor depende de la secuencia exacta de intercambio de procesos y del tamaño de dichos procesos.

Particionamiento

- Las particiones fijas y dinámicas son ineficientes en el uso de la memoria generan fragmentación interna o externa.
- Por esta razón se han desarrollado técnicas más eficientes como la paginación y la segmentación que veremos a continuación.

Contenido I

Memoria Virtual

- Permite que el tamaño del programa, datos y la pila combinados pueden ser mayores que la memoria disponible para ese proceso.

Memoria Virtual

- Permite que el tamaño del programa, datos y la pila combinados pueden ser mayores que la memoria disponible para ese proceso.
- El sistema operativo guarda aquellas partes del programa de uso corriente en la memoria principal y el resto en disco.

Memoria Virtual

- Permite que el tamaño del programa, datos y la pila combinados pueden ser mayores que la memoria disponible para ese proceso.
- El sistema operativo guarda aquellas partes del programa de uso corriente en la memoria principal y el resto en disco.
- El sistema operativo decide que partes del programa puede cargar en memoria, cuando y donde ubicarlas, corriendo el riesgo de perder demasiado tiempo en acceso al disco si la elección no es correcta.

Memoria Virtual

- Permite que el tamaño del programa, datos y la pila combinados pueden ser mayores que la memoria disponible para ese proceso.
- El sistema operativo guarda aquellas partes del programa de uso corriente en la memoria principal y el resto en disco.
- El sistema operativo decide que partes del programa puede cargar en memoria, cuando y donde ubicarlas, corriendo el riesgo de perder demasiado tiempo en acceso al disco si la elección no es correcta.
- **Ventajas:** permite tener mas programas cargados a la vez, posibilidad de trabajar con programas de mayor tamaño que la memoria físicas, y la reducción de la frecuencia de intercambio entre procesos.

Contenido I

Paginación

- En esta técnica la memoria principal se divide en porciones de tamaño fijo relativamente pequeñas y cada proceso también se divide en porciones del mismo tamaño.

Paginación

- En esta técnica la memoria principal se divide en porciones de tamaño fijo relativamente pequeñas y cada proceso también se divide en porciones del mismo tamaño.
- A las porciones del programa se las conoce como **páginas** y las porciones de memoria se las conoce como **marcos de página**.

Paginación

- En esta técnica la memoria principal se divide en porciones de tamaño fijo relativamente pequeñas y cada proceso también se divide en porciones del mismo tamaño.
- A las porciones del programa se las conoce como **páginas** y las porciones de memoria se las conoce como **marcos de página**.
- Esta técnica genera una fragmentación interna muy pequeña (solo una fracción de la última página del proceso) y no genera fragmentación externa.

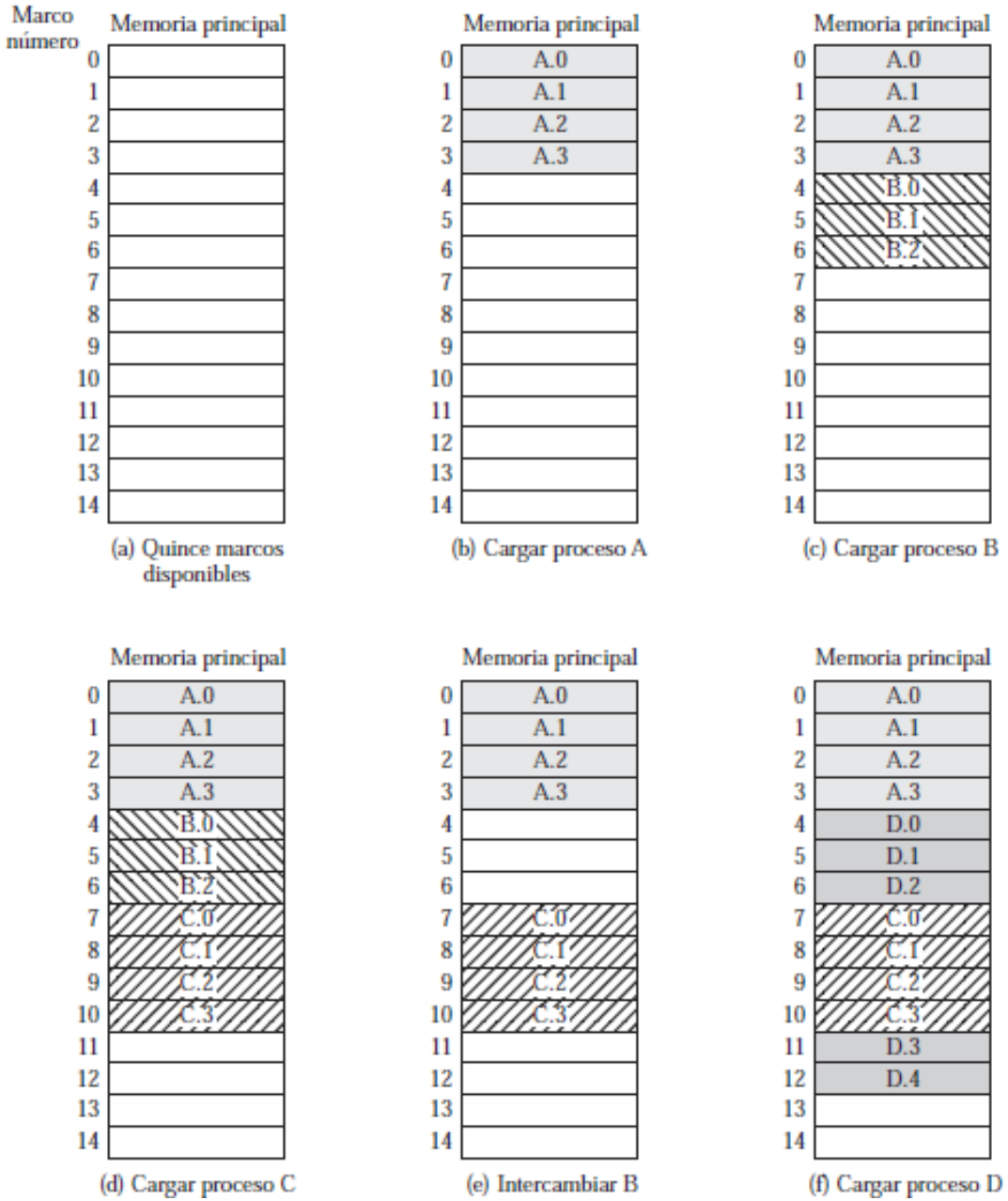
Paginación

- En esta técnica la memoria principal se divide en porciones de tamaño fijo relativamente pequeñas y cada proceso también se divide en porciones del mismo tamaño.
- A las porciones del programa se las conoce como **páginas** y las porciones de memoria se las conoce como **marcos de página**.
- Esta técnica genera una fragmentación interna muy pequeña (solo una fracción de la última página del proceso) y no genera fragmentación externa.
- En un momento dado algunos marcos estarán en uso y otros disponibles. El SO mantiene una tabla de marcos libres.

Paginación

- En esta técnica la memoria principal se divide en porciones de tamaño fijo relativamente pequeñas y cada proceso también se divide en porciones del mismo tamaño.
- A las porciones del programa se las conoce como **páginas** y las porciones de memoria se las conoce como **marcos de página**.
- Esta técnica genera una fragmentación interna muy pequeña (solo una fracción de la última página del proceso) y no genera fragmentación externa.
- En un momento dado algunos marcos estarán en uso y otros disponibles. El SO mantiene una tabla de marcos libres.
- Se producen los **fallos de página** .

Paginación



- Proceso B se suspende y es intercambiado.
- Proceso D toma los marcos que dejo B y dos adicionales.

Paginación

Tabla de páginas

- Una tabla de páginas contiene una entrada por cada página del proceso (el índice de la tabla indica el número de página).

Paginación

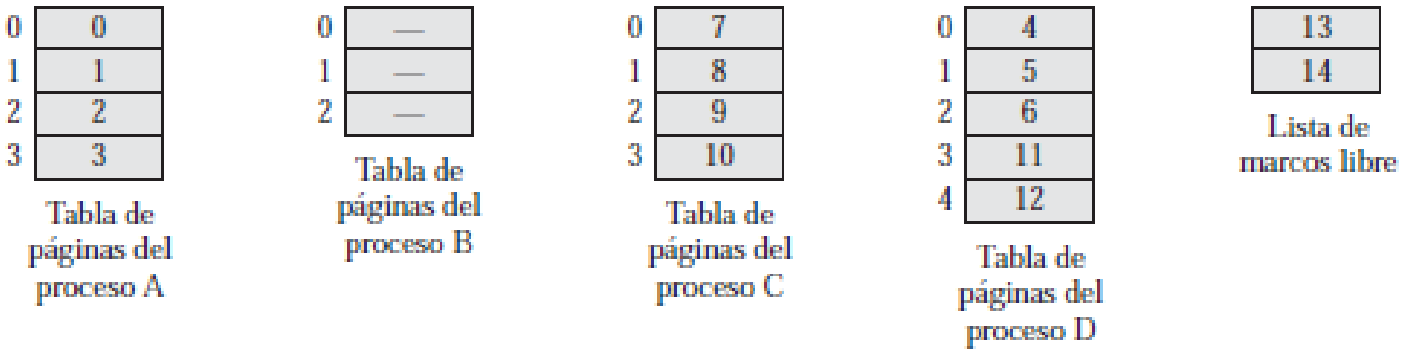
Tabla de páginas

- Una tabla de páginas contiene una entrada por cada página del proceso (el índice de la tabla indica el número de página).
- Cada entrada de la tabla de páginas contiene el número del marco en la memoria principal que contiene la página correspondiente.

Paginación

Tabla de páginas

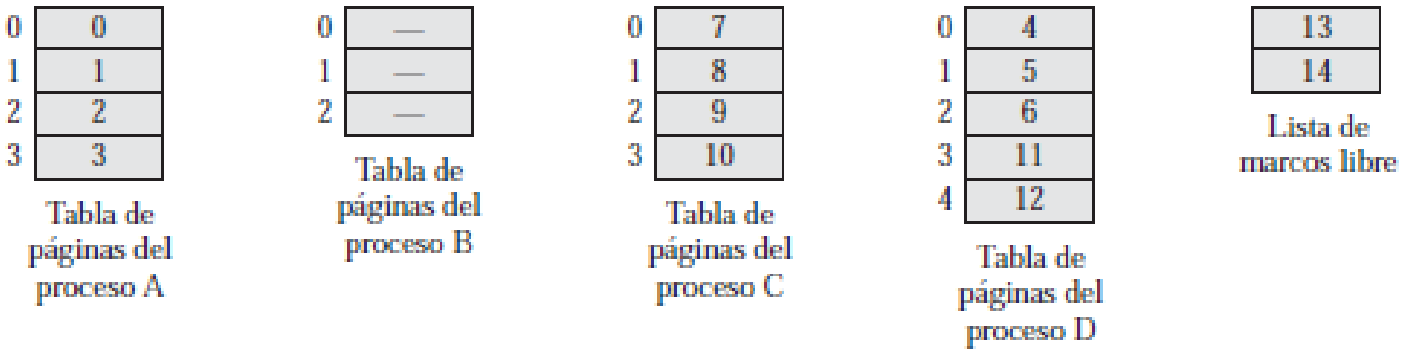
- Una tabla de páginas contiene una entrada por cada página del proceso (el índice de la tabla indica el número de página).
- Cada entrada de la tabla de páginas contiene el número del marco en la memoria principal que contiene la página correspondiente.
- El SO mantiene una lista de marcos libres.



Paginación

Tabla de páginas

- Una tabla de páginas contiene una entrada por cada página del proceso (el índice de la tabla indica el número de página).
- Cada entrada de la tabla de páginas contiene el número del marco en la memoria principal que contiene la página correspondiente.
- El SO mantiene una lista de marcos libres.



- Vemos que la paginación simple es similar al particionamiento fijo, pero las particiones son bastante pequeñas; un programa podría ocupar más de una partición; y dichas particiones no necesitan ser contiguas.

Algoritmos de reemplazo de páginas

- **El óptimo:** cuál sería el algoritmo de reemplazo de páginas óptimo?

  . Lo malo es que es imposible de implementar por que no se puede determinar cuantas instrucciones se ejecutarán antes de hacer referencia a cada una de las páginas.

Algoritmos de reemplazo de páginas

- **El óptimo:** cuál sería el algoritmo de reemplazo de páginas óptimo? reemplazar a la página a la que se hará referencia en el futuro más lejano. Lo malo es que es imposible de implementar por que no se puede determinar cuantas instrucciones se ejecutarán antes de hacer referencia a cada una de las páginas.

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.
  - Clase 1: no ha sido referenciada, ha sido modificada.

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.
  - Clase 1: no ha sido referenciada, ha sido modificada.
  - Clase 2: ha sido referenciada, no ha sido modificada.

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.
  - Clase 1: no ha sido referenciada, ha sido modificada.
  - Clase 2: ha sido referenciada, no ha sido modificada.
  - Clase 3: ha sido referenciada, ha sido modificada.

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.
  - Clase 1: no ha sido referenciada, ha sido modificada.
  - Clase 2: ha sido referenciada, no ha sido modificada.
  - Clase 3: ha sido referenciada, ha sido modificada.
- Cuales se desearia reemplazar?

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.
  - Clase 1: no ha sido referenciada, ha sido modificada.
  - Clase 2: ha sido referenciada, no ha sido modificada.
  - Clase 3: ha sido referenciada, ha sido modificada.
- Cuales se desearia reemplazar? reemplaza al azar una de las clases con menor numeración

No usadas recientemente

Algoritmos de reemplazo de páginas

- Se dividen las páginas en 4 categorias:
  - Clase 0: no ha sido referenciada, no ha sido modificada.
  - Clase 1: no ha sido referenciada, ha sido modificada.
  - Clase 2: ha sido referenciada, no ha sido modificada.
  - Clase 3: ha sido referenciada, ha sido modificada.
- Cuales se desearia reemplazar? reemplaza al azar una de las clases con menor numeración
- Facil de entender, eficiente de implementar y buen rendimiento.

FIFO

Algoritmos de reemplazo de páginas

- No es la mejor solución, no toma en cuenta que es lo que se está eliminando.
- Ejemplo: Un supermercado que reemplaza los productos de sus repisas con esta política puede eliminar el gel de cabello o un producto como la harina, arroz, agua.
- Lo mismo ocurre con las páginas y procesos.

Menos usadas recientemente

Algoritmos de reemplazo de páginas

- Se basa en el principio de localidad. Las páginas que se han utilizado recientemente tienen alta probabilidad de ser referenciadas a corto plazo.
- Por esto reemplaza las páginas menos usadas recientemente, es decir la que tenga más tiempo sin ser referenciada.

Contenido I

Segmentación

- Esta técnica subdivide un programa de usuario y sus datos asociados en segmentos.

Segmentación

- Esta técnica subdivide un programa de usuario y sus datos asociados en segmentos.
- No es necesario que todos los programas sean de la misma longitud pero existe una longitud máxima de segmento.

Segmentación

- Esta técnica subdivide un programa de usuario y sus datos asociados en segmentos.
- No es necesario que todos los programas sean de la misma longitud pero existe una longitud máxima de segmento.
- debido al uso de segmentos de distinto tamaño, la segmentación es similar al particionamiento dinámico pero en este caso un programa puede ocupar más de una partición y estas particiones no tienen que ser contiguas.

Segmentación

- Esta técnica subdivide un programa de usuario y sus datos asociados en segmentos.
- No es necesario que todos los programas sean de la misma longitud pero existe una longitud máxima de segmento.
- debido al uso de segmentos de distinto tamaño, la segmentación es similar al particionamiento dinámico pero en este caso un programa puede ocupar más de una partición y estas particiones no tienen que ser contiguas.
- Elimina la fragmentación interna pero sufre de fragmentación externa (pero en menor cantidad).

Segmentación

- Esta técnica subdivide un programa de usuario y sus datos asociados en segmentos.
- No es necesario que todos los programas sean de la misma longitud pero existe una longitud máxima de segmento.
- debido al uso de segmentos de distinto tamaño, la segmentación es similar al particionamiento dinámico pero en este caso un programa puede ocupar más de una partición y estas particiones no tienen que ser contiguas.
- Elimina la fragmentación interna pero sufre de fragmentación externa (pero en menor cantidad).
- Contrario a la paginación, esta técnica es visible al programador y se proporciona como una utilidad para organizar programas y datos.

Segmentación

- Esta técnica subdivide un programa de usuario y sus datos asociados en segmentos.
- No es necesario que todos los programas sean de la misma longitud pero existe una longitud máxima de segmento.
- debido al uso de segmentos de distinto tamaño, la segmentación es similar al particionamiento dinámico pero en este caso un programa puede ocupar más de una partición y estas particiones no tienen que ser contiguas.
- Elimina la fragmentación interna pero sufre de fragmentación externa (pero en menor cantidad).
- Contrario a la paginación, esta técnica es visible al programador y se proporciona como una utilidad para organizar programas y datos.
- El programador debe asignar programas y datos a diferentes segmentos por lo tanto debe conocer la limitación de tamaño de segmento.

Segmentación

- Debido a los distintos tamaños de segmentos no existe una relación directa entre direcciones lógicas y físicas como en la paginación.

Segmentación

- Debido a los distintos tamaños de segmentos no existe una relación directa entre direcciones lógicas y físicas como en la paginación.
- Un esquema de segmentación sencillo haría uso de una **tabla de segmentos** por cada proceso y una **lista de bloques libre** de memoria principal.

Segmentación

- Debido a los distintos tamaños de segmentos no existe una relación directa entre direcciones lógicas y físicas como en la paginación.
- Un esquema de segmentación sencillo haría uso de una **tabla de segmentos** por cada proceso y una **lista de bloques libre** de memoria principal.
- Cada entrada de la tabla de segmentos tendría que proporcionar la dirección inicial de la memoria principal del correspondiente segmento y la longitud del segmento, para asegurar que no se utilizan direcciones no válidas.

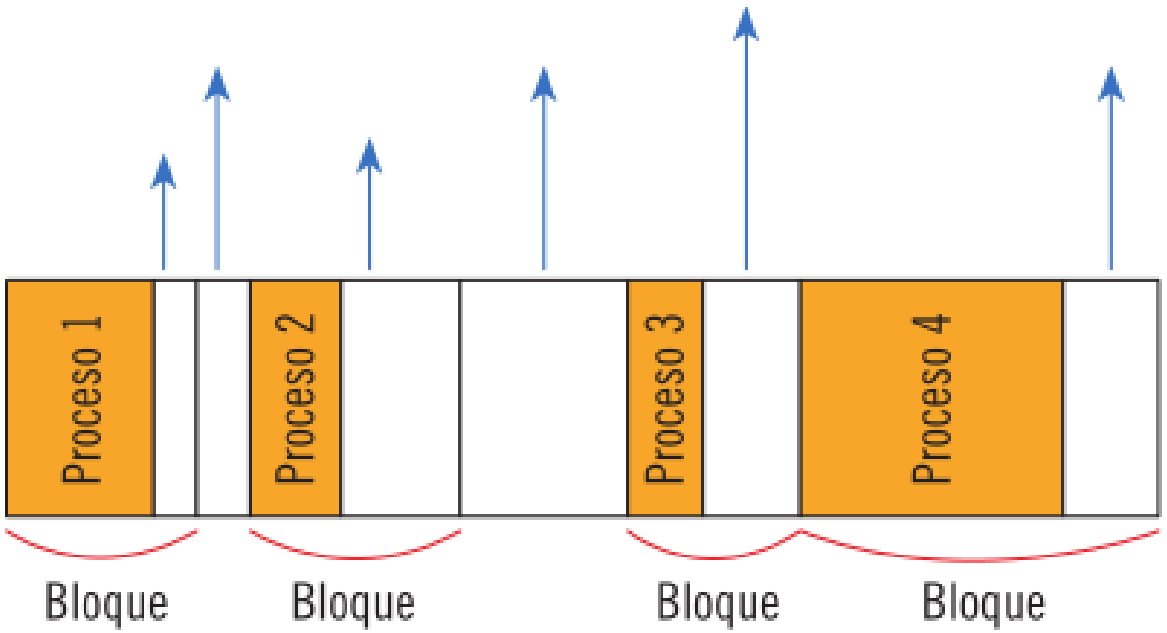
Segmentación

- Debido a los distintos tamaños de segmentos no existe una relación directa entre direcciones lógicas y físicas como en la paginación.
- Un esquema de segmentación sencillo haría uso de una **tabla de segmentos** por cada proceso y una **lista de bloques libre** de memoria principal.
- Cada entrada de la tabla de segmentos tendría que proporcionar la dirección inicial de la memoria principal del correspondiente segmento y la longitud del segmento, para asegurar que no se utilizan direcciones no válidas.
- Cuando un proceso entra en el estado Ejecutando, la dirección de su tabla de segmentos se carga en un registro especial utilizado por el hardware de gestión de la memoria.

Contenido I

Ejercicio

Indique el tipo de partición que corresponde en cada espacio:



Ejercicio

La tabla muestra tres páginas con el tiempo en el que fue cargada, la última referencia, y los bits de referecia (R) y Modificada (M)

Página	Cargada	Última referencia	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0

a ¿Cuál página reemplazará el algoritmo NRU?

b ¿Cuál página reemplazará el algoritmo FIFO?

c ¿Cuál página reemplazará el algoritmo LRU?