

Sistemas Operativos

Administración de la memoria

Clase 7 - Unidad III

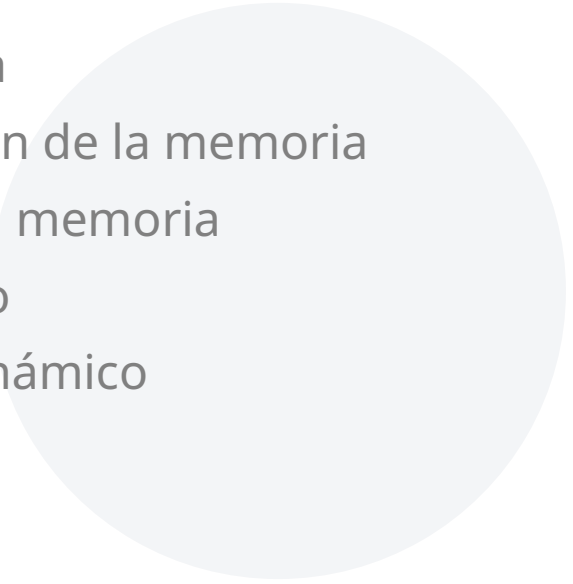
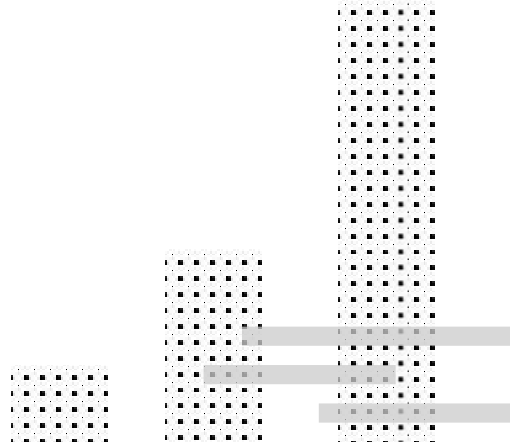
Lic. Alexis Sostersich
sostersich.alexis@uader.edu.ar



Teoría – Sistemas Operativos

Administración de la memoria

Unidad III – Clase 7

- Gestión de la memoria
 - Requisitos de la gestión de la memoria
 - Particionamiento de la memoria
 - Particionamiento fijo
 - Particionamiento dinámico
- 
- 

Teoría – Sistemas Operativos

Administración de la memoria

En un sistema monoprogramado, la memoria se divide en dos partes: una parte para el sistema operativo (monitor residente, núcleo) y una parte para el programa actualmente en ejecución.

En un sistema multiprogramado, la parte de «usuario» de la memoria se debe subdividir posteriormente para acomodar múltiples procesos.

El sistema operativo es el encargado de la tarea de subdivisión y a esta tarea se le denomina gestión de la memoria.

Teoría – Sistemas Operativos

Administración de la memoria

Una gestión de la memoria efectiva es vital en un sistema multiprogramado.

Si solo unos pocos procesos se encuentran en memoria, entonces durante una gran parte del tiempo todos los procesos esperarían por operaciones de E/S y el procesador estaría ocioso.

Es necesario asignar la memoria para asegurar una cantidad de procesos listos que consuman el tiempo de procesador disponible.

Teoría – Sistemas Operativos

Requisitos de la gestión de la memoria

Estos son requisitos que la gestión de la memoria debe satisfacer:

- Reubicación.
- Protección.
- Compartición.
- Organización lógica.
- Organización física.

Teoría – Sistemas Operativos

Reubicación

En un sistema multiprogramado, la memoria principal disponible se comparte entre varios procesos. Normalmente, no es posible que el programador sepa qué programas residirán en memoria principal.

Adicionalmente, sería bueno poder intercambiar procesos en la memoria principal para maximizar la utilización del procesador, proporcionando un gran número de procesos para la ejecución.

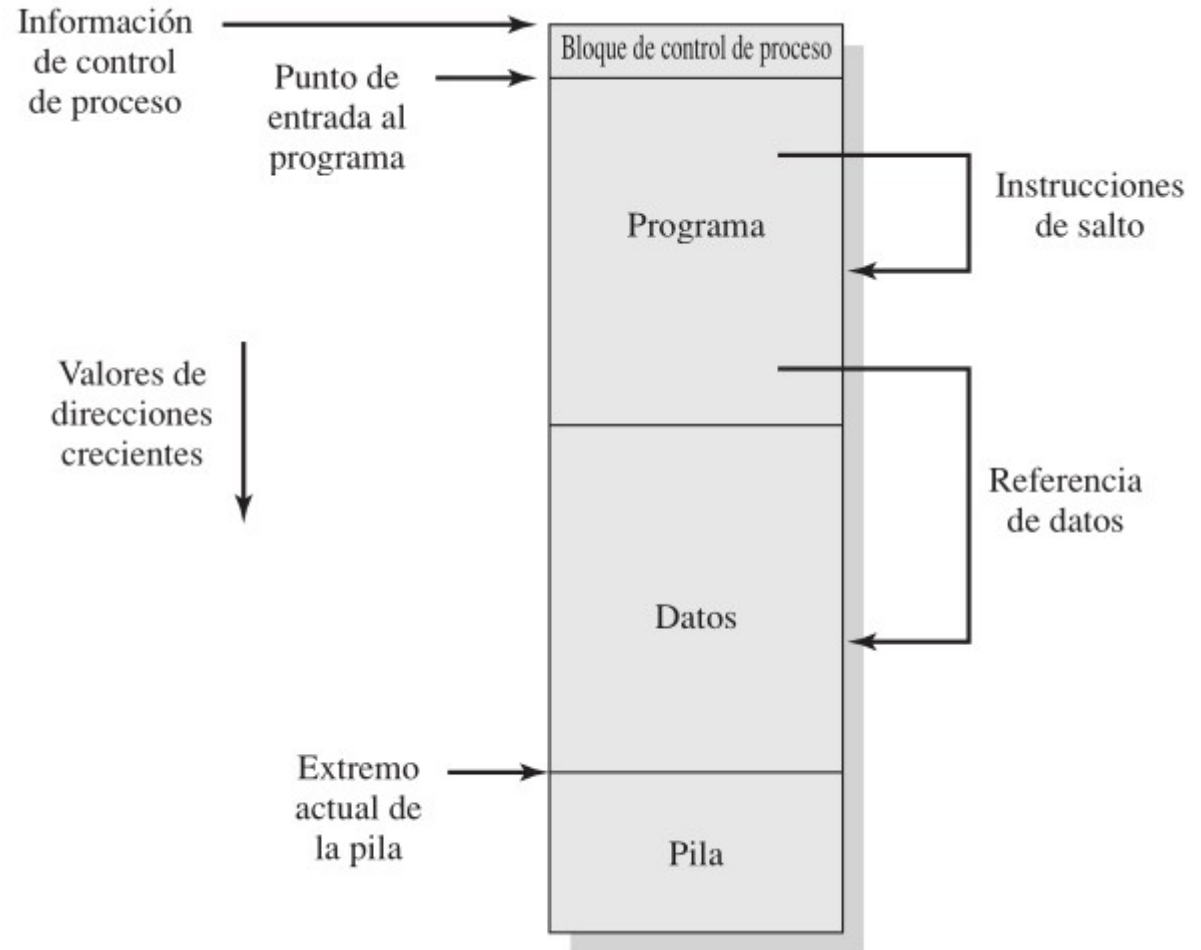
Una vez que un programa se ha llevado al disco, sería bastante limitante tener que colocarlo en la misma región de memoria principal donde se hallaba anteriormente, cuando vuelva a la memoria. Por el contrario, podría ser necesario reubicar el proceso a un área de memoria diferente.

Teoría – Sistemas Operativos

Reubicación

No se puede conocer de forma anticipada dónde se va a colocar un programa y se debe permitir que los programas se puedan mover en la memoria principal, debido al intercambio o swap.

El procesador debe tratar con referencias de memoria dentro del propio programa.



Teoría – Sistemas Operativos

Protección

Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos, sean accidentales o intencionadas.

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.

Lograr los requisitos de la reubicación incrementa la dificultad de satisfacer los requisitos de protección. Más aún, la mayoría de los lenguajes de programación permite el cálculo dinámico de direcciones en tiempo de ejecución. Por tanto, todas las referencias de memoria generadas por un proceso deben comprobarse en tiempo de ejecución para poder asegurar que se refieren solo al espacio de memoria asignado a dicho proceso.

Los requisitos de protección de memoria deben ser satisfechos por el procesador (hardware) en lugar del sistema operativo (software).

Teoría – Sistemas Operativos

Compartición

Cualquier mecanismo de protección debe tener la flexibilidad de permitir a varios procesos acceder a la misma porción de memoria principal.

Por ejemplo, si varios programas están ejecutando el mismo programa, es ventajoso permitir que cada proceso pueda acceder a la misma copia del programa en lugar de tener su propia copia separada. Procesos que estén cooperando en la misma tarea podrían necesitar compartir el acceso a la misma estructura de datos. Por tanto, el sistema de gestión de la memoria debe permitir el acceso controlado a áreas de memoria compartidas sin comprometer la protección esencial.

Veremos que los mecanismos utilizados para dar soporte a la reubicación soportan también capacidades para la compartición.

Teoría – Sistemas Operativos

Organización lógica

Casi invariablemente, la memoria principal de un computador se organiza como un espacio de almacenamiento lineal o unidimensional, compuesto por una secuencia de bytes o palabras.

A nivel físico, la memoria secundaria está organizada de forma similar. Mientras que esta organización es similar al hardware real de la máquina, no se corresponde a la forma en la cual los programas se construyen normalmente.

La mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (solo lectura, solo ejecución) y algunos de los cuales contienen datos que se pueden modificar.

Teoría – Sistemas Operativos

Organización lógica

Si se pueden tratar de forma efectiva los programas de usuarios y los datos en la forma de módulos de algún tipo, logramos varias 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 sobrecarga adicional modesta, 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.

Teoría – Sistemas Operativos

Organización física

La memoria del computador se organiza en al menos dos niveles, conocidos como memoria principal y memoria secundaria.

La memoria principal proporciona acceso rápido a un coste relativamente alto. Adicionalmente, la memoria principal es volátil; es decir, no proporciona almacenamiento permanente.

La memoria secundaria es más lenta y más barata que la memoria principal y normalmente no es volátil. Por tanto, la memoria secundaria de larga capacidad puede proporcionar almacenamiento para programas y datos a largo plazo, mientras que una memoria principal más pequeña contiene programas y datos actualmente en uso.

Teoría – Sistemas Operativos

Organización física

En este esquema de dos niveles, la organización del flujo de información entre la memoria principal y secundaria supone una de las preocupaciones principales del sistema. La responsabilidad para este flujo podría asignarse a cada programador en particular, pero no es practicable o deseable por dos motivos:

1. La memoria principal disponible para un programa más sus datos podría ser insuficiente. El programador debería utilizar una técnica de superposición (overlaying), en la cual los programas y los datos se organizan de tal forma que se puede asignar la misma región de memoria a varios módulos.
2. En un entorno multiprogramado, el programador no conoce en tiempo de codificación cuánto espacio estará disponible o dónde se localizará dicho espacio.

Por tanto, está claro que la tarea de mover la información entre los dos niveles de la memoria debería ser una responsabilidad del sistema. Esta tarea es la esencia de la gestión de la memoria.

Teoría – Sistemas Operativos

Particionamiento de la memoria

La operación principal de la gestión de la memoria es traer los procesos a la memoria principal para que el procesador los pueda ejecutar. En casi todos los sistemas multiprogramados modernos, esto implica el uso de un esquema sofisticado denominado memoria virtual. Por su parte, la memoria virtual se basa en una o ambas de las siguientes técnicas básicas: segmentación y paginación.

Antes de fijarse en estas técnicas de memoria virtual, se debe preparar el camino, analizando técnicas más sencillas que no utilizan memoria virtual.

Una de estas técnicas, el particionamiento, se ha utilizado en algunas variantes de ciertos sistemas operativos ahora obsoletos. Las otras dos técnicas, paginación sencilla y segmentación sencilla, no son utilizadas de forma aislada.

Teoría – Sistemas Operativos

Particionamiento fijo

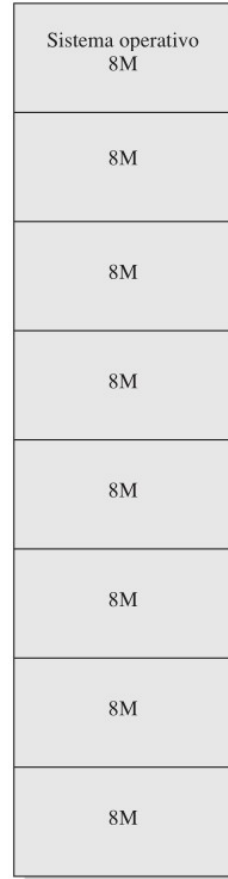
En la mayoría de los esquemas para gestión de la memoria, se puede asumir que el sistema operativo ocupa alguna porción fija de la memoria principal y que el resto de la memoria principal está disponible para múltiples procesos. El esquema más simple para gestionar la memoria disponible es repartirla en regiones con límites fijos.

Tamaños de partición

Una posibilidad consiste en hacer uso de particiones 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.

Teoría – Sistemas Operativos

Particionamiento fijo de una memoria de 64 Mb



(a) Participaciones de igual tamaño



(b) Participaciones de distinto tamaño

Teoría – Sistemas Operativos

Particionamiento fijo

Existen dos dificultades con el uso de las particiones fijas del mismo tamaño:

- Un programa podría ser demasiado grande para caber en una partición. En este caso, el programador debe diseñar el programa con el uso de overlays, de forma que sólo se necesite una porción del programa en memoria principal en un momento determinado. Cuando se necesita un módulo que no está presente, el programa de usuario debe cargar dicho módulo en la partición del programa, superponiéndolo (overlaying) a cualquier programa o datos que haya allí.
- La utilización de la memoria principal es extremadamente ineficiente. Cualquier programa, sin importar lo pequeño que sea, ocupa una partición entera. Por ejemplo, podría haber un programa cuya longitud es menor que 2 Mbytes; ocuparía una partición de 8 Mbytes cuando se lleva a la memoria. Este fenómeno, en el cual hay espacio interno malgastado debido al hecho de que el bloque de datos cargado es menor que la partición, se conoce con el nombre de fragmentación interna.

Teoría – Sistemas Operativos

Particionamiento fijo: Algoritmo de ubicación

Con particiones del mismo tamaño, la ubicación de los procesos en memoria es trivial. En cuanto haya una partición disponible, un proceso se carga en dicha partición. Debido a que todas las particiones son del mismo tamaño, no importa qué partición se utiliza. Si todas las particiones se encuentran ocupadas por procesos que no están listos para ejecutar, entonces uno de dichos procesos debe llevarse a disco para dejar espacio para un nuevo proceso. Cuál de los procesos se lleva a disco es una decisión de planificación.

Con particiones de diferente tamaño, hay dos formas posibles de asignar los procesos a las particiones. La forma más sencilla consiste en asignar cada proceso a la partición más pequeña dentro de la cual cabe. En este caso, se necesita una cola de planificación para cada partición, que mantenga procesos en disco destinados a dicha partición. La ventaja de esta técnica es que los procesos siempre se asignan de tal forma que se minimiza la memoria malgastada dentro de una partición (fragmentación interna).

Teoría – Sistemas Operativos

Particionamiento fijo

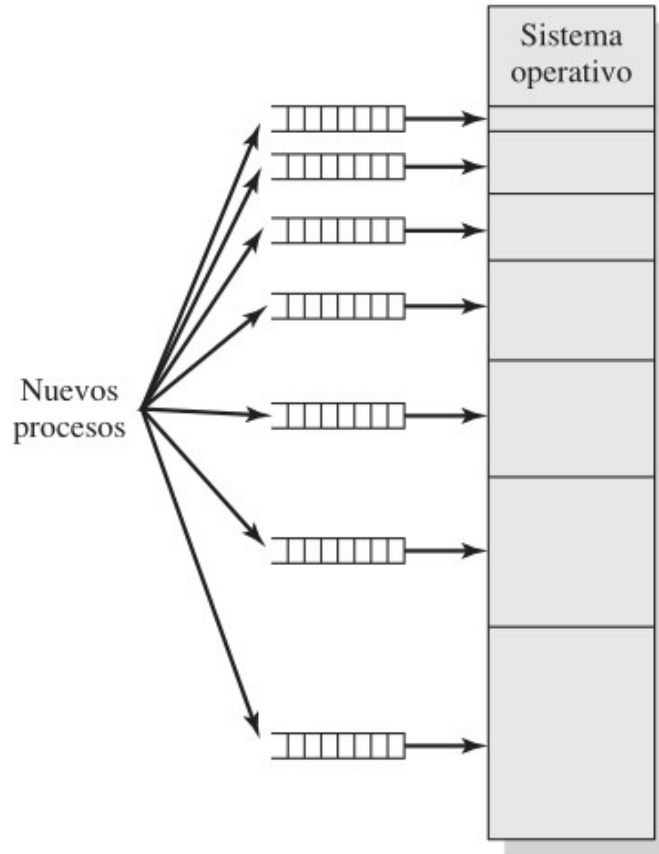
El uso de particiones de distinto tamaño proporciona un grado de flexibilidad frente a las particiones fijas. Adicionalmente, se puede decir que los esquemas de particiones fijas son relativamente sencillos y requieren un soporte mínimo por parte del sistema operativo y una sobrecarga de procesamiento mínimo. Sin embargo, tiene una serie de desventajas:

- El número de particiones especificadas en tiempo de generación del sistema limita el número de procesos activos (no suspendidos) del sistema.
- Debido a que los tamaños de las particiones son preestablecidos en tiempo de generación del sistema, los trabajos pequeños no utilizan el espacio de las particiones eficientemente.

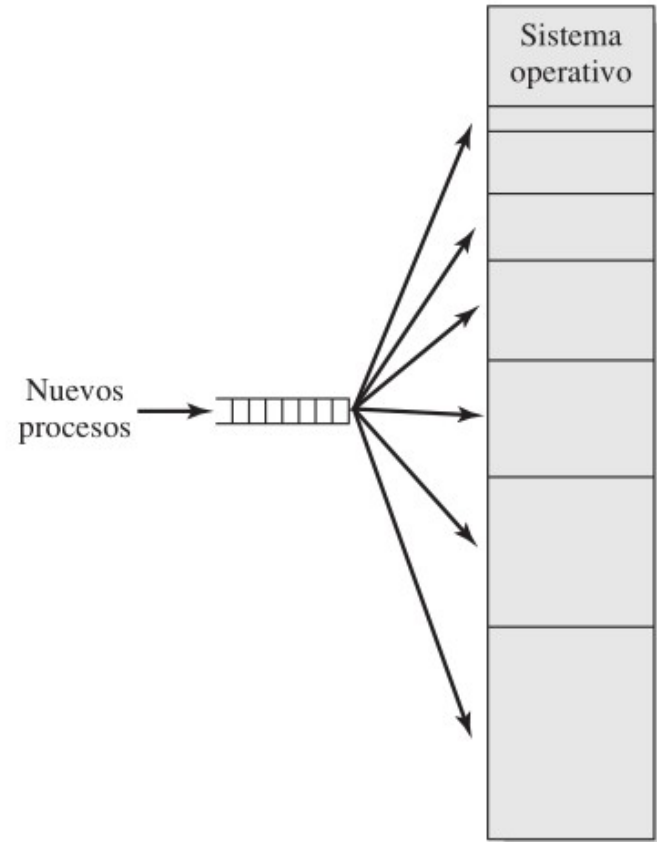
El uso de particionamiento fijo es casi desconocido hoy en día. Un ejemplo de un sistema operativo exitoso que sí utilizó esta técnica fue un sistema operativo de los primeros mainframes de IBM, el sistema operativo OS/MFT (Multiprogramming with a Fixed Number of Tasks; Multiprogramado con un número fijo de tareas).

Teoría – Sistemas Operativos

Asignación de memoria para particionamiento fijo



(a) Una cola de procesos por participación



(b) Un única cola

Teoría – Sistemas Operativos

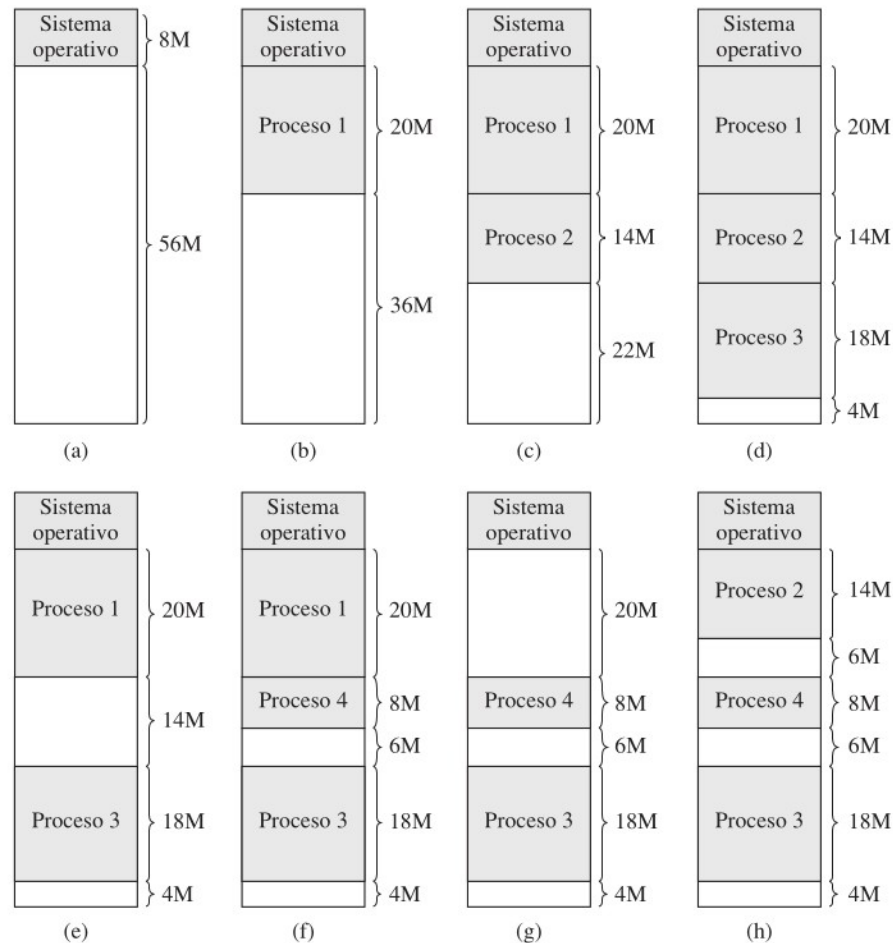
Particionamiento dinámico

Para vencer algunas de las dificultades con particionamiento fijo, se desarrolló una técnica conocida como particionamiento dinámico. De nuevo, esta técnica se ha sustituido por técnicas de gestión de memoria más sofisticadas. Un sistema operativo importante que utilizó esta técnica fue el sistema operativo de mainframes de IBM, el sistema operativo OS/MVT (Multiprogramming with a Variable Number of Tasks; Multiprogramado con un número variable de tareas).

Con particionamiento dinámico, las particiones son de longitud y número variable. Cuando se lleva un proceso a la memoria principal, se le asigna exactamente tanta memoria como requiera y no más.

Teoría – Sistemas Operativos

El efecto del particionamiento dinámico



Teoría – Sistemas Operativos

Particionamiento dinámico

El método comienza correctamente, pero finalmente lleva a una situación en la cual existen muchos huecos pequeños en la memoria. A medida que pasa el tiempo, la memoria se fragmenta cada vez más y la utilización de la memoria se decrementa. Este fenómeno se conoce como fragmentación externa, indicando que la memoria que es externa a todas las particiones se fragmenta de forma incremental, por contraposición a lo que ocurre con la fragmentación interna.

Una técnica para eliminar la fragmentación externa es la compactación: de vez en cuando, el sistema operativo desplaza los procesos en memoria, de forma que se encuentren contiguos y de este modo toda la memoria libre se encontrará unida en un bloque.

La desventaja de la compactación es que consume tiempo de procesador. Observe que la compactación requiere la capacidad de reubicación dinámica. Es decir, debe ser posible mover un programa desde una región a otra en la memoria principal sin invalidar las referencias de la memoria de cada programa.

Teoría – Sistemas Operativos

Particionamiento dinámico: Algoritmo de ubicación

Debido a que la compactación de memoria consume una gran cantidad de tiempo, el diseñador del sistema operativo debe ser inteligente a la hora de decidir cómo asignar la memoria a los procesos (cómo eliminar los huecos). A la hora de cargar o intercambiar un proceso a la memoria principal, y siempre que haya más de un bloque de memoria libre de suficiente tamaño, el sistema operativo debe decidir qué bloque libre asignar.

Tres algoritmos de colocación que pueden considerarse son mejor-ajuste (best-fit), primer-ajuste (first-fit) y siguiente-ajuste (next-fit). Todos, por supuesto, están limitados a escoger entre los bloques libres de la memoria principal que son iguales o más grandes que el proceso que va a llevarse a la memoria. Mejor-ajuste escoge el bloque más cercano en tamaño a la petición. Primer-ajuste comienza a analizar la memoria desde el principio y escoge el primer bloque disponible que sea suficientemente grande. Siguiente-ajuste comienza a analizar la memoria desde la última colocación y elige el siguiente bloque disponible que sea suficientemente grande.

Teoría – Sistemas Operativos

Particionamiento dinámico: Algoritmo de reemplazamiento

En un sistema multiprogramado que utiliza particionamiento dinámico, puede haber un momento en el que todos los procesos de la memoria principal estén en estado bloqueado y no haya suficiente memoria para un proceso adicional, incluso después de llevar a cabo una compactación. Para evitar malgastar tiempo de procesador esperando a que un proceso se desbloquee, el sistema operativo intercambiará alguno de los procesos entre la memoria principal y disco para hacer sitio a un nuevo proceso o para un proceso que se encuentre en estado Listo-Suspendido. Por tanto, el sistema operativo debe escoger qué proceso reemplazar. Debido a que el tema de los algoritmos de reemplazo se contemplará en detalle respecto a varios esquemas de la memoria virtual, se pospone esta discusión hasta entonces.

Teoría – Sistemas Operativos

Sistema Buddy

Ambos esquemas de particionamiento, fijo y dinámico, tienen desventajas. El particionamiento fijo limita el número de procesos activos y puede utilizar el espacio ineficientemente si existe un mal ajuste entre los tamaños de partición disponibles y los tamaños de los procesos. El particionamiento dinámico es más complejo de mantener e incluye la sobrecarga de la compactación. Un compromiso interesante es el sistema buddy, donde los bloques de memoria disponibles son de tamaño 2^K , $L \leq K \leq U$, donde:

2^L = bloque de tamaño más pequeño asignado

2^U = bloque de tamaño mayor asignado; normalmente $2U$ es el tamaño de la memoria completa disponible.

Para comenzar, el espacio completo disponible se trata como un único bloque de tamaño 2^U . Si se realiza una petición de tamaño s , tal que $2^{U-1} < s \leq 2^U$, se asigna el bloque entero. En otro caso, el bloque se divide en dos bloques buddy iguales de tamaño 2^{U-1} . Si $2^{U-2} < s \leq 2^{U-1}$, entonces se asigna la petición a uno de los otros dos bloques.

Teoría – Sistemas Operativos

Sistema Buddy

En otro caso, uno de ellos se divide por la mitad de nuevo. Este proceso continúa hasta que el bloque más pequeño mayor o igual que s se genera y se asigna a la petición. En cualquier momento, el sistema buddy mantiene una lista de huecos (bloques sin asignar) de cada tamaño 2^i . Un hueco puede eliminarse de la lista $(i+1)$ dividiéndolo por la mitad para crear dos bloques de tamaño 2^i en la lista i . Siempre que un par de bloques de la lista i no se encuentren asignados, son eliminados de dicha lista y unidos en un único bloque de la lista $(i+1)$. Si se lleva a cabo una petición de asignación de tamaño k tal que $2^{i-1} < k \leq 2^i$, se utiliza el siguiente algoritmo recursivo para encontrar un hueco de tamaño 2^i :

El sistema buddy sirve para eliminar las desventajas de ambos esquemas fijo y variable, pero en los sistemas operativos contemporáneos, la memoria virtual basada en paginación y segmentación es superior. Sin embargo, el sistema buddy se ha utilizado en sistemas paralelos como una forma eficiente de asignar y liberar programas paralelos. Una forma modificada del sistema buddy se utiliza en la asignación de memoria del núcleo UNIX.

Bibliografía

- Tanenbaum, A. S. (2009). Sistemas operativos modernos (3a ed.) (pp. 1-18). México: Pearson Educación.
- Stallings, W. (2005). Sistemas operativos (5a ed.) (pp. 54-67). Madrid: Pearson Educación.

