MEMORIA

En un sist mono programado .la memoria se divide en 2 (SO y otra para el programa en ejecución), en uno multiprogramado la parte del usuario se debe dividir para acomodar los múltiples procesos

El SO es el encargado de la subdivisión (esta tarea se llama gestión de memoria muy importante para que el procesador no este ocioso).

Los requisitos de la gestión de memoria son:

**Reubicación:**

Es importante poder mover la ubicación de la memoria para que esto lno entorpezca el proceso(luego de volver del disco), por esto se reubica en un lugar diferente

**Protección**:

c/u de los procesos debe poder protegerse contra interferencias no deseadas de otros procesos (no deben ser capaces de referenciarse sin permisos )(un proceso no puede interferir al de otro sin un trato especial)

**Compartición**

Cualquier mecanismo de protección debe tener la flexibilidad de permitir a varios procesos acceder a la misma porción de memoria principal. Por tanto, el sistema de gestión de la memoria debe permitir el acceso controlado a áreas de memoria compartidas sin comprometer la protección esencial.

**Org. Lógica:**

son direcciones de memoria donde están esos datos, ya sea de memoria ram, disco o memoria virtual

**Org física:**

es el orden en que están guardados los datos de forma física

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

**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.

Para no usar la memoria virtual en sistemas obsoletos una de las técnicas son el particionado

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 es repartirla en regiones con límites fijos.

**Tamaños de partición**

**Fijas**

Las particiones pueden ser fijas(dificultades si la partición es muy grande y quedar ocioso , se puede usar overlays para arreglarlo. Otra dificultad es que la utilización de memoria ppal es ineficiente por que el la partición es muy grande para la instrucción a ejecutar ej necesita 2mb y usa una partición de 8mb , se conoce con el nombre de **fragmentación interna** )

Solución parcial del problema

**Dinámicas**

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.

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, descrita anteriormente.

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.

**Ver si entras los algoritmos de ubicación.**

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 ala 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

**Paginación**

memoria principal se divide en porciones de tamaño fijo relativamente pequeños, y que cada proceso también se divide en porciones pequeñas del mismo tamaño fijo. A dichas porciones del ceso, conocidas como páginas, se les asigna porciones disponibles de memoria, conocidas como marcos, o marcos de páginas.

Esta sección muestra que el espacio de memoria malgastado por cada proceso debido a la fragmentación interna corresponde sólo a una fracción de la última página de un proceso. **No existe fragmentación externa.**

La tabla de páginas muestra la ubicación del marco por cada página del proceso. Dentro del programa, cada dirección lógica está formada por un número de página y un desplazamiento dentro de la página. Por tanto vemos que la paginación simple, tal y como se describe aquí, es similar al particionamiento fijo. Las diferencias son que, con la paginación, las particiones son bastante pequeñas; un programa podría ocupar más de una partición; y dichas particiones no necesitan ser contiguas. Para hacer este esquema de paginación conveniente, el tamaño de página y por tanto el tamaño del marco debe ser una potencia de 2.

los siguientes pasos para la traducción de direcciones:

• Extraer el número de página como los n bits de la izquierda de la dirección lógica.

•Utilizar el número de página como un índice a tabla de páginas del proceso para encontrar el número de marco, k.

• La dirección física inicial del marco es k ¥ 2m, y la dirección física del byte referenciado es dicho número más el desplazamiento. Esta dirección física no necesita calcularse, es fácilmente construida concatenando el número de marco al desplazamiento.

**Segmentación**

Un programa de usuario se puede subdividir utilizando segmentación, en la cual el programa y susdatos asociados se dividen en un número de segmentos. No se requiere que todos los programas seande la misma longitud, aunque hay una longitud máxima de segmento.

No tiene fragmentación interna, pero si tiene externa(aunque es mucho menor a la part dinámica)

Visible al programador. Utiliza una tabla de segmentos para ubcar los datos ya que no usa ni la ubicación física ni lógica.

Como busca:

considérese una dirección de n+m bits, donde los n bits de la izquierda corresponden al número

de segmento y los m bits de la derecha corresponden al desplazamiento. En el ejemplo (Figura 7.11c),

n = 4 y m = 12. Por tanto, el tamaño de segmento máximo es 212 = 4096. Se necesita llevar a cabo los siguientes pasos para la traducción de direcciones

-Extraer el número de segmento como los n bits de la izquierda de la dirección lógica

-Utilizar el número de segmento como un índice a la tabla de segmentos del proceso para encontrar la dirección física inicial del segmento

Comparar el desplazamiento, expresado como los m bits de la derecha, y la longitud del segmento.

Si el desplazamiento es mayor o igual que la longitud, la dirección no es válida.

• La dirección física deseada es la suma de la dirección física inicial y el desplazamiento

**Carga absoluta**

un cargador absoluto requiere que un módulo de carga dado debe cargarse siempre en la misma ubicación de la memoria principal. r, todas las referencias a direcciones deben ser direcciones de memoria principal específicas o absoluta(ubicación exacta en memoria)

**Carga reubicable**

Mas rápido que el anterior debido a que puede ubicarse en cualquier lugar de la mem ppal , logrando un mejor tiempo de carga.( utiliza direcciones relativas a algún punto conocido, tal como el inicio del programa)no es recomendable para memoria virtual

**Carga dinámica en tiempo real**

el módulo de carga se carga en la memoria principal con todas las referencias de la memoria en formato relativo . Hasta que una instrucción no se ejecuta realmente, no se calcula la dirección absoluta. Se utiliza un software para no perjudicar el rendimiento del hardware.

**ENLACE**

La función de un montador es tomar como entrada una colección de módulos objeto y producir un módulo de carga, formado por un conjunto integrado de programa y módulos de datos, que se pasará al cargado

**Editor de enlace**

La naturaleza de este enlace de direcciones dependerá del tipo de módulo de carga que se cree y cuando se lleve a cabo el proceso de enlace

Para **el enlace dinámico** **en tiempo de carga**, se deben seguir los siguientes pasos. El módulo de carga (módulo de aplicación) debe llevarse a memoria para cargarlo. Cualquier referencia a un módulo externo (módulo destino) provoca que el cargador encuentre al módulo destino, lo cargue y altere la referencia a una dirección relativa a la memoria

Ventajas: se facilita las versiones modificadas o actualizadas del modulo destino, al ser en un fichero de este tipo de enlace se puede compartir el código de forma automática

Con **enlace dinámico** **en tiempo de ejecución**, algunos de los enlaces son pospuestos hasta el tiempo de ejecución. Las referencias externas a los módulos destino quedan en el programa cargado. Cuando se realiza una llamada a un módulo ausente, el sistema operativo localiza el módulo, lo carga y lo enlaza al módulo llamante.

**Memoria virtual**

Memoria Virtual: Lo uso de RAM. Por lo tanto, puede trabajar con grandes procesos ya que en la RAM solo carga una parte. El mayor beneficio, es que puede ejecutar procesos más grandes que el tamaño de la memoria RAM.

Conjunto Residente: Es una pequeña porción de la RAM que se asigna a cada proceso, que incluye la porción inicial del programa y de datos sobre la cual acceden las primeras instrucciones. En este se intenta hacer un equilibrio entre la cantidad de páginas, los procesos, tiempo de E/L, etc. El tamaño del conjunto residente hace referencia a la cantidad de procesos.

Reubicación: Reubica los procesos dentro de la RAM cuando dicho proceso no está siendo usado.

Principio de proximidad: Indica que las referencias al programa y a los datos dentro de un proceso, tienden a agruparse. Por lo tanto, se resume que solo unas pocas porciones del proceso se necesitaran a lo largo de un periodo de tiempo corto .

Buffer de traducción anticipada (TLB): Es una memoria pequeña de alta velocidad estilo cache que contiene aquellas entradas de la tabla de páginas que han sido usadas de forma más reciente. Funcionamiento: Dada una dirección virtual, el procesador primero examina primero la TLB, si la entrada de la tabla de páginas solicitada está presente (acierto en TLB), entonces se recupera el número de marco y se construye la dirección real. Si existe un fallo en la TLB, el procesador utiliza el número de página para indexar la tabla de páginas del procesador y examinar la correspondiente entrada de la tabla de páginas.

La incorporación de la **memoria virtual a la paginación y segmentación** beneficia en la capacidad de poder de ejecutar más procesos y/o procesos más pesados/complejos. Esto es gracias a que en la memoria principal solo se almacenan los conjuntos residentes de todos los procesos, y no toda la información de cada proceso.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

En la mayoría de sistemas, existe una única tabla de página por proceso. Pero cada proceso puede ocupar una gran cantidad de memoria, por lo que la mayoría de los esquemas, almacena las tablas de página en la memoria virtual utilizando la **TLB**.

Existe una **tabla de páginas invertidas**, la cual es una estrategia alternativa al uso de la tabla de páginas de uno o varios niveles. Lo único que se requiere para estas tablas de página siempre es una proporción fija de la memoria real, independientemente del número de procesos o de las páginas virtuales soportadas. Esta contiene número de página, identificador del proceso, bits de control y puntero de la cadena (un valor hash).

Diagrama

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media En todos los casos debemos lograr un **tamaño de página** equilibrado para no tener demasiada fragmentación interna, ni demasiadas páginas ya que si tenemos un mayor número de páginas por proceso, significa que también mayores **tablas de páginas**. Por lo tanto, en procesos grandes, la tabla de páginas debe encontrarse en la memoria virtual, eso hace que pueda haber un **fallo de página** doble para una referencia sencilla a memoria, el primero para atraer la tabla de páginas a la parte solicitada y el segundo para atraer la página del propio proceso.

En la **segmentacion con memoria virtual** tenemos varias ventajas para el programados, las cuales son:

* Simplifica el tratamiento de estructuras de datos que pueden crecer, de forma que si no se conoce el tamaño exacto, se puede hacer una aproximación.
* Permite programas que se modifiquen o recopilen de forma independiente, sin requerir que el conjunto completo de programas se re-enlacen.
* Da soporte a la compartición entre procesos.
* Soporta los mecanismos de protección, esto es porque el programador puede asignar privilegios de acceso de una forma apropiada.

**Paginación y segmentación combinadas**

Diagrama

Descripción generada automáticamenteEn un sistema combinado de paginación/segmentación, el espacio de direcciones del usuario se divide en un número de segmentos, a discreción del programador. Cada segmento es dividido en un número de páginas de tamaño fijo, que son del tamaño de los marcos de la memoria principal. Si un segmento tiene longitud inferior a una página, el segmento ocupará únicamente una página. Desde el punto de vista del sistema, el desplazamiento dentro del segmento es visto como un número de página y un desplazamiento dentro de la página incluida en el segmento.

La segmentación proporciona una vía para la implementación de las **políticas de protección y compartición**, debido a que cada entrada en la tabla de segmentos incluye la longitud y dirección base, por lo que un programa no puede, de forma descontrolada, acceder a una posición de memoria principal más allá de los límites del segmento (protección). Por lo tanto, para poder tener compartición, es posible que un segmento este referenciado desde las tabas de segmento de más de un proceso.

Cuando el tamaño de la memoria principal crece, la proximidad de referencia de las aplicaciones va a decrecer.

Software del Sistema Operativo

El diseño de la parte de la gestión de memoria depende de tres opciones fundamentales:

1. Si el sistema usa o no técnicas de memoria virtual.
2. El uso de paginación o segmentación o ambas.
3. Los algoritmos utilizados para los diferentes aspectos de la gestión de memoria.

Además, el sistema operativo tiene políticas para la memoria virtual, explicadas en la siguiente tabla.

Tabla

Descripción generada automáticamente con confianza media

* + - **Política de recuperación:** determina cuando una página se trae a la memoria principal.
      * Bajo Demanda: Una página se trae a memoria solo cuando se hace referencia a una posición en dicha página.
      * Paginación adelantada: Trae otras páginas además de la solicitada. Si las paginas están almacenadas en memoria secundaria de forma secundaria, es más eficiente traer varias. Obvio que si dichas páginas no se referían, es ineficiente.
    - **Política de Ubicación**: Determina en que parte de la memoria real van a residir las porciones de la memoria de un proceso.
    - **Política de Reemplazo:** Es la selección de una página en la memoria principal como cantidad para reemplazarse cuando se va a traer una nueva página.
      * Algoritmos básicos.
        + **Optimo:** Se tomara como reemplazo la página para la cual el instante de la siguiente referencia se encuentra más lejos. No sirve, el futuro no se puede ver y si apago la PC se pierde.
        + **FIFO**: Saca el que más tiempo lleva en la fila.
        + **LRU** (usada menos frecuentemente): Saco la que no se haya referenciado desde hace más tiempo, debido al principio de proximidad referenciada.
        + **De Reloj**: Agrega un bit de usado, este se coloca en 1 cada vez que es referenciado, utilizado con un buffer circular. De esta forma se reemplaza el que tenga el bit en 0 mientras el buffer va recorriendo el círculo.
        + **Buffers de Página**: Se agrega una lista, de forma que cuando se reemplaza una página, esta no se elimina sino que se guarda ese marco de memoria y se añade al final de la lista de páginas libres. De esta forma, si se hace referencia a esa página, se devuelve el conjunto residente en la memoria.
    - **Gestión del conjunto residente.**
      * Tamaño: El SO debe reservar memoria principal para cada proceso.
        + **Fijo**: Se decide en el momento inicial de la carga del proceso.
        + **Variable:** Reserva un numero de marcos por proceso que puede variar a lo largo del tiempo de vida del mismo.
      * Ámbito de reemplazo: Se activa por medio de un fallo de página cuando no existen marcos de página libre.
        + **Global:** Considera todas las páginas de la memoria principal.
        + Interfaz de usuario gráfica, Texto

          Descripción generada automáticamente**Local**: Selecciona entre las paginas residentes del proceso que genero el fallo.
    - **Política de Limpieza**: Se encarga de determinar cuándo una página que esta modificada se debe escribir en la memoria secundaria. Lo opuesto a la política de recuperación.
      * Bajo Demanda: Se escribe cuando se seleccione para un reemplazo
      * Limpieza adelantada: Se escriben las páginas modificadas antes de que sus marcos de páginas se necesiten, de forma que las páginas se puedan escribir en lotes.
    - **Control de carga**: Determina el número de procesos que residirán en la memoria principal.
      * Grado de multiprogramación: Es el máximo punto de utilización del procesador. Para poder reducir dicho número debemos **SUSPENDER PROCESOS.**
        + **Procesos con baja prioridad** (es por políticas de activación y no rendimiento).
        + **Proceso que provoca muchos fallos.**
        + **Proceso activado hace más tiempo.**
        + **Proceso con el conjunto residente de mejor tamaño.**
        + **Proceso mayor o más grande.**
        + **Proceso con la mayor ventana de ejecución restante.**