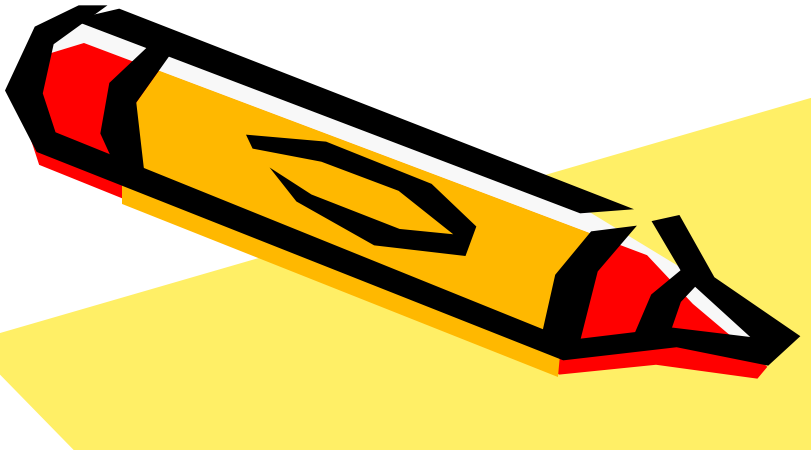
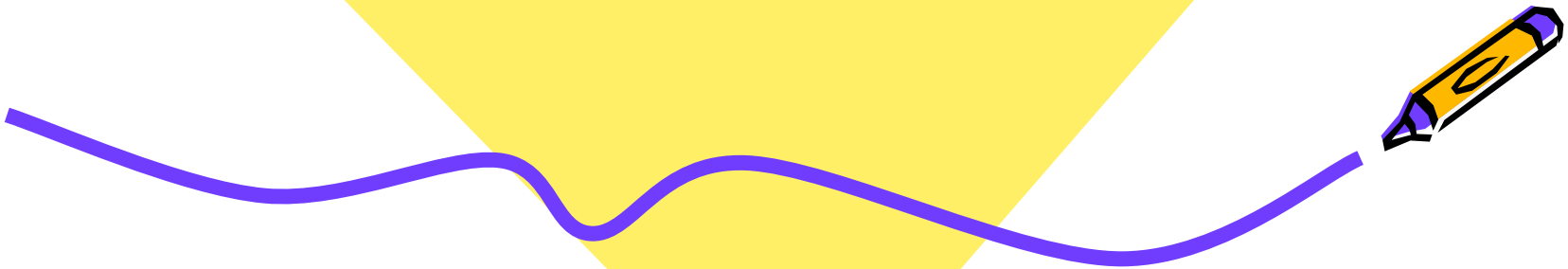


memoria

Unidad V



Contenido

- Listar los mecanismos para administrar la memoria
- Explicar los mecanismos de administración de la memoria
- Explicar el intercambio de memoria
- Explicar el uso de la memoria virtual

Introducción

- En un sistema que no es multitarea, la memoria principal se divide en dos partes:
  - Una para el sistema operativo (monitor residente, kernel)
  - Otra parte para el programa que se ejecuta en ese instante.

Introducción

- En un sistema multitarea, la parte de "usuario" debe subdividirse aun mas para hacer sitio a varios procesos.
- La tarea de subdivisión la lleva a cabo dinámicamente el sistema operativo y se conoce como **administración de memoria**.

Introducción

- En un sistema multitarea resulta vital una administración efectiva de la memoria.
- Si solo hay unos pocos procesos en memoria, entonces la mayor parte del tiempo estarán esperando a la E/S y el procesador estará desocupado.

Introducción

- Por ello, hace falta repartir eficientemente la memoria para meter tantos procesos como sea posible.

Objetivos administración de memoria

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

Reubicación

- Generalmente un programador desconoce previamente que otros programas residirán en memoria en el momento de la ejecución de su programa.

Reubicación

- Se busca poder cargar y descargar los procesos activos en la memoria principal para maximizar el uso del procesador, manteniendo una gran reserva de procesos listos para ejecutar.

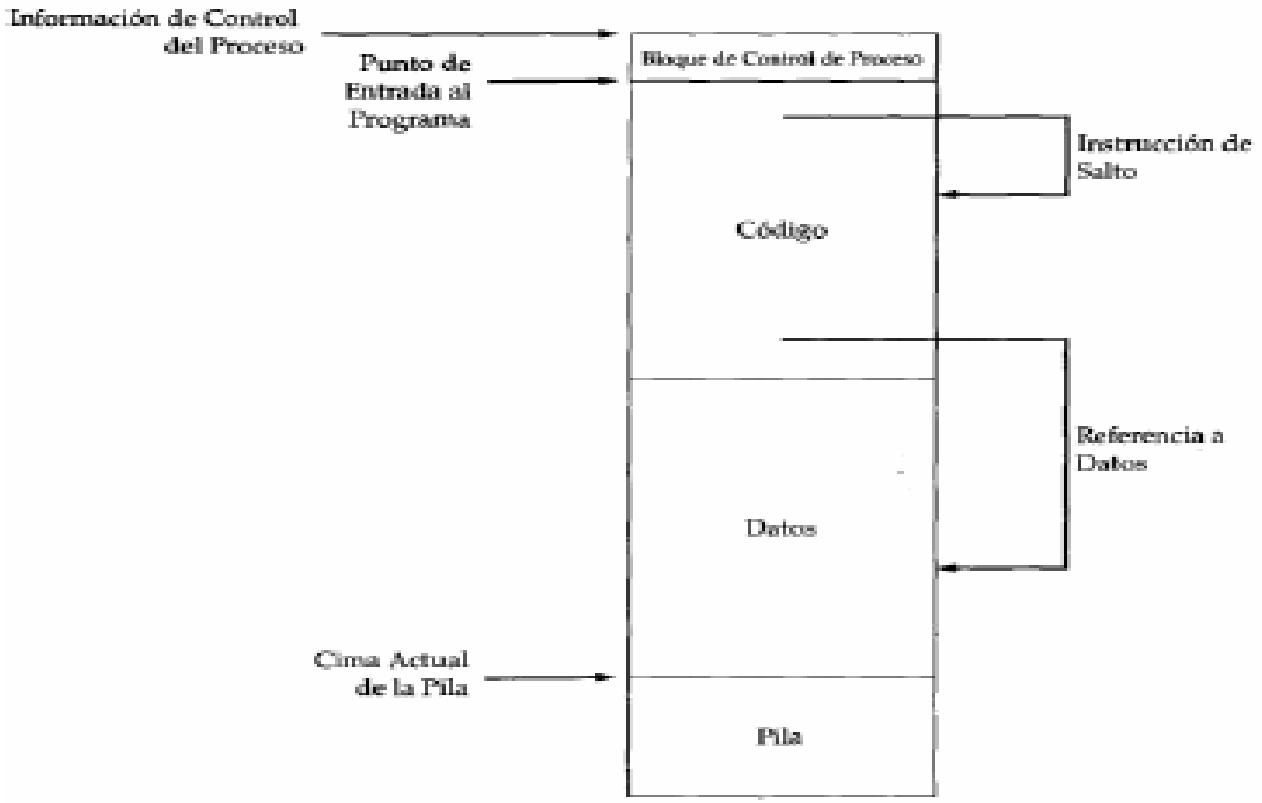
Reubicación

- La descarga se realiza a una partición del disco denominada swap.
- El programa que haya sido descargado al disco, se limitara a declarar que, cuando vuelva a ser cargado, en lo posible sea a la misma región de memoria principal que antes.

Reubicación

- De este modo, se sabe antes de tiempo donde debe situarse un programa y hay que permitir que el programa pueda moverse en memoria principal como resultado de un intercambio

Reubicación



Reubicación

- Se supondrá que la imagen del proceso ocupa una región contigua de la memoria principal.
- Se tiene que conocer la ubicación de la información de control del proceso y de la pila de ejecución

Reubicación

- También el punto de partida para comenzar la ejecución del programa para dicho proceso.
- El sistema operativo maneja la memoria y es responsable de traer el proceso a memoria principal, las direcciones deben ser fáciles de conseguir.

Reubicación

- El procesador debe ocuparse de las referencias a memoria dentro del programa.
- Las instrucciones de bifurcación deben contener la dirección que haga referencia a la instrucción que se vaya a ejecutar a continuación.

Reubicación

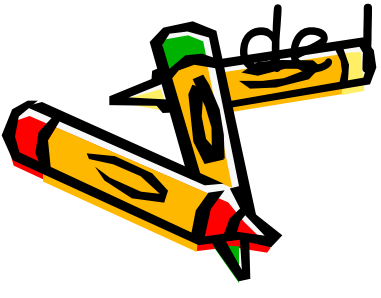
- Las instrucciones que hagan referencia a datos deben contener la dirección del byte o de la palabra de datos referenciada.

Reubicación

- El hardware del procesador y el software del sistema operativo deben traducir las referencias a memoria encontradas en el código del programa a las direcciones físicas reales que reflejen la posición actual del programa en memoria principal.

Protección

- Cada proceso debe protegerse contra interferencias no deseadas de otros procesos, tanto accidentales como intencionadas.
- El código de un proceso no puede hacer referencia a posiciones de memoria de otros procesos, con fines de lectura o escritura, sin permiso.



Protección

- Hasta cierto punto, satisfacer las exigencias de reubicación aumenta la dificultad de satisfacción de las exigencias de protección.

Protección

- Se desconoce la ubicación de un programa en memoria principal, es imposible comprobar las direcciones absolutas durante la compilación para asegurar la protección.

Protección

- La mayoría de los lenguajes de programación permiten el calculo dinámico de direcciones durante la ejecución, generando, por ejemplo, un índice de un vector o un apuntador a una estructura de datos.

Protección

- Todas las referencias a memoria generadas por un proceso deben comprobarse durante la ejecución para asegurar que solo hacen referencia al espacio de memoria destinado a dicho proceso.

Protección

- Los mecanismos que respaldan la reubicación también forman parte básica del cumplimiento de las necesidades de protección

Protección

- Normalmente, un proceso de usuario no puede acceder a ninguna parte del sistema operativo, tanto programa como datos.

Protección

- El programa de un proceso no puede en general bifurcar hacia una instrucción de otro proceso.
- Sin un acuerdo especial, el programa de un proceso no puede acceder al área de datos de otro proceso.
- El procesador debe ser capaz de abandonar tales instrucciones en el momento de la ejecución.

Protección

- Las exigencias de protección de memoria pueden ser satisfechas por el procesador (hardware) en vez de por el sistema operativo (software).

Protección

- El sistema operativo no puede anticiparse a todas las referencias a memoria que hará un programa.
- Si fuera posible, seria prohibitivo en términos de tiempo consumido el proteger cada programa por adelantado de posibles violaciones de referencias a memoria.

Protección

- Solo es posible evaluar la tolerancia de una referencia a memoria (acceso a datos o bifurcación) durante la ejecución de la instrucción que realiza la referencia.
- Para llevar esto a cabo, el hardware del procesador debe poseer dicha capacidad.

Compartición

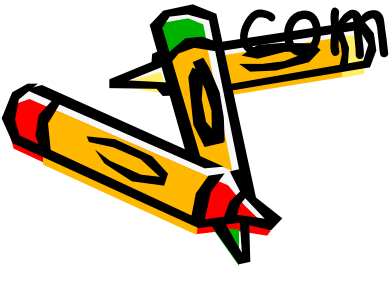
- Cualquier mecanismo de protección que se implemente debe tener la flexibilidad de permitir el acceso de varios procesos a la misma zona de memoria principal.

Compartición

- Si una serie de procesos están ejecutando el mismo programa, resultaría beneficioso permitir a cada proceso que acceda a la misma copia del programa, en lugar de tener cada uno su propia copia

Compartición

- Los procesos que cooperan en una tarea pueden necesitar acceso compartido a la misma estructura de datos.
- La administración de memoria debe permitir accesos controlados a las áreas compartidas de la memoria, sin comprometer la protección básica.



Organización Lógica

- De forma casi invariable, la memoria principal de un sistema informático se organiza como un espacio de direcciones lineal o unidimensional que consta de una secuencia de bytes o palabras.

Organización Lógica

- La memoria secundaria, a nivel físico, se organiza de forma similar.
- Si bien esta organización refleja fielmente el hardware de la maquina, no se corresponde con la forma en la que los programas están construidos habitualmente.

Organización Lógica

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

Organización Lógica

- Si el sistema operativo y el hardware pueden tratar de forma efectiva los programas de usuario y los datos en forma de módulos de algún tipo, se conseguirá una serie de ventajas:

Organización Lógica

- 1. Los módulos pueden escribirse y compilarse independientemente, mientras que el sistema resuelve durante la ejecución todas las referencias de un modulo a otro.

Organización Lógica

- 2. Con un bajo precio adicional, pueden otorgarse varios grados de protección (solo lectura, solo ejecución) a los distintos módulos.

Organización Lógica

- 3. Es posible introducir mecanismos por medio de los cuales los procesos puedan compartir módulos.

Organización Física

- La memoria se organiza en, al menos, dos niveles: memoria principal y memoria secundaria.

Organización Física

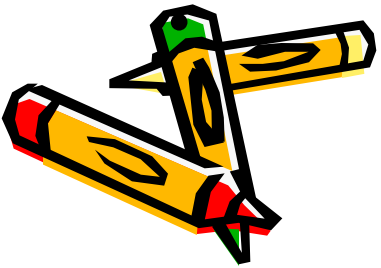
- La memoria principal ofrece un acceso rápido con un precio relativamente alto.
- Además, la memoria principal es volátil; esto es, no proporciona almacenamiento permanente.

Organización Física

- La memoria secundaria es mas lenta y barata que la memoria principal, tiene mucho mayor capacidad y, normalmente, no es volátil.

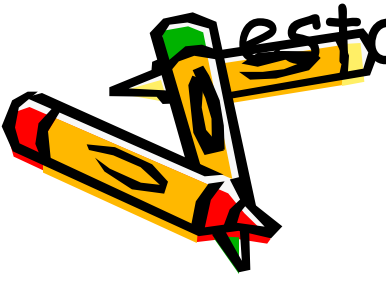
Organización Física

- De este modo, una memoria secundaria de gran capacidad puede permitir un almacenamiento a largo plazo de programas y datos, al tiempo que una memoria principal pequeña mantiene los programas y datos de uso actual.



Organización Física

- En este esquema a 2 niveles, la organización del flujo de información entre la memoria principal y secundaria tiene gran interés en el sistema.
- La responsabilidad de este flujo podría asignarse al programador, pero esto es impracticable e indeseable



Organización Física

- La memoria principal disponible para un programa y sus datos puede ser insuficiente.
- El programador no conoce durante la codificación cuanto espacio habrá disponible o donde estará este espacio.

Organización Física

- La tarea de mover información entre los dos niveles de memoria debe ser responsabilidad del sistema.
- Esta tarea es la esencia de la administración de memoria.

Herramientas básicas de manejo de memoria

- La paginación, cada proceso se divide en paginas de tamaño constante y relativamente pequeño.
- La segmentación permite el uso de partes de tamaño variable.
- También es posible combinar la segmentación y la paginación en un único esquema de manejo de memoria.

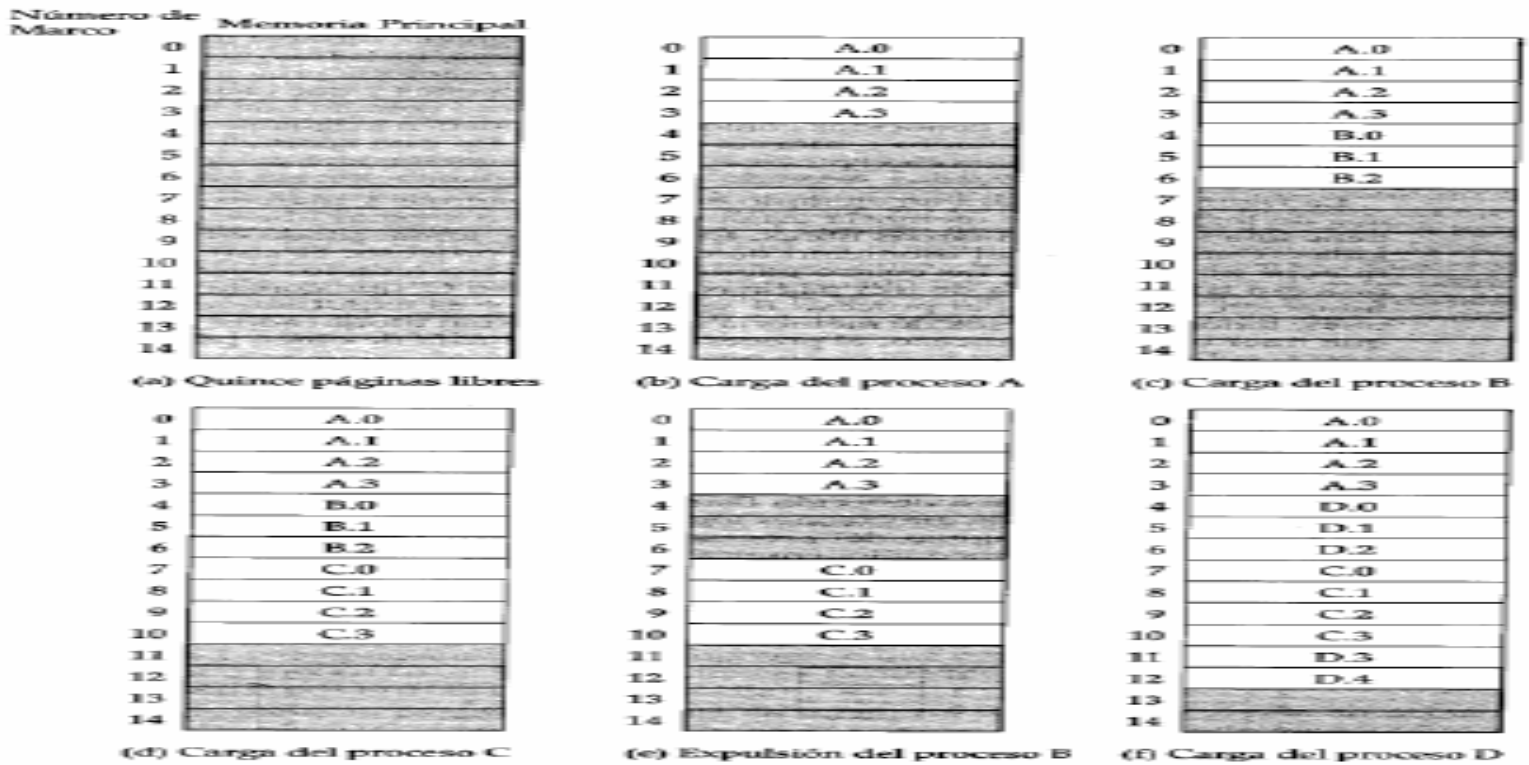
Paginación Simple

- Se supone, que la memoria principal se encuentra particionada en trozos iguales de tamaño fijo relativamente pequeños y que cada proceso esta dividido también en pequeños trozos de tamaño fijo y del mismo tamaño que los de memoria.

Paginación Simple

- Los trozos del proceso, conocidos como **páginas, pueden asignarse a los trozos libres de** memoria, conocidos como marcos o **marcos de pagina.**

Paginación Simple



Paginación Simple

- La lámina anterior muestra un ejemplo del uso de paginas y marcos.
- En un instante dado, algunos de los marcos de memoria están en uso y otros están libres.
- El sistema operativo mantiene una lista de los marcos libres.

Paginación Simple

- El proceso A, almacenado en disco, consta de cuatro paginas.
- Cuando llega el momento de cargar este proceso, el sistema operativo busca cuatro marcos libres y carga las cuatro paginas del proceso A en los cuatro marcos

Paginación Simple

- El proceso B, que consta de tres paginas y el proceso C, que consta de cuatro, se cargan a continuación.
- Mas tarde, el proceso B se suspende y es expulsado de memoria principal.

Paginación Simple

- Después, todos los procesos de memoria principal están bloqueados y el sistema operativo tiene que traer un nuevo proceso, el proceso D, que consta de cinco paginas.

Paginación Simple

- Supóngase ahora, como en este ejemplo, que no hay suficientes marcos sin usar contiguos para albergar al proceso.
- ¿Impedirá esto al sistema operativo cargar D?

Paginación Simple

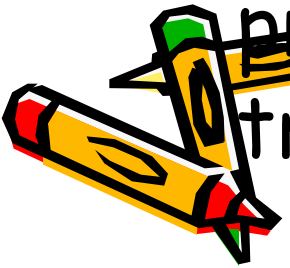
- La respuesta es no, puesto que se puede emplear de el concepto de dirección lógica.
- Ya no será suficiente con un simple registro base.

Paginación Simple

- En su lugar, el sistema operativo mantiene una **tabla de páginas para** cada proceso.
- La tabla de paginas muestra la posición del marco de cada pagina del proceso.

Paginación Simple

- Dentro del programa, cada dirección lógica constara de un numero de pagina y de un desplazamiento dentro de la pagina.
- En el caso de la partición simple, una dirección lógica era la posición de una palabra relativa al comienzo del programa; el procesador realizaba la traducción a dirección física.



Paginación Simple

- Con paginación, el hardware del procesador también realiza la traducción de direcciones lógicas a físicas.
- Ahora, el procesador debe saber como acceder a la tabla de paginas del proceso actual.

Paginación Simple

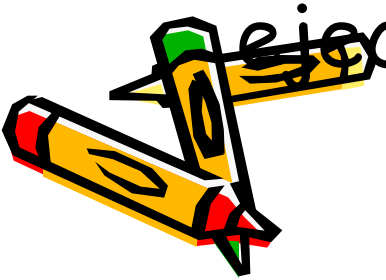
- Dada una dirección lógica (numero de pagina, desplazamiento), el procesador emplea la tabla de paginas para obtener una dirección física (numero de marco, desplazamiento).

Paginación Simple

- Convine usar un tamaño de pagina que sea potencia de dos.
- El esquema de direccionamiento lógico es transparente al programador, al ensamblador y al montador.

Paginación Simple

- Cada dirección lógica de un programa (numero de pagina, desplazamiento) es idéntica a su dirección relativa.
- También, resulta relativamente sencillo realizar una función hardware para llevar a cabo la traducción de direcciones dinámicas durante la ejecución.



Paginación Simple

- Resumiendo, mediante la paginación simple, la memoria principal se divide en pequeños marcos del mismo tamaño.

Paginación Simple

- Cada proceso se divide en paginas del tamaño del marco; los procesos pequeños necesitaran pocas paginas, mientras que los procesos grandes necesitaran mas.

Paginación Simple

- Cuando se introduce un proceso en memoria, se cargan todas sus paginas en los marcos libres y se rellena su tabla de paginas.
- Esta técnica resuelve la mayoría de los problemas inherentes a la partición.

Segmentación Simple

- En este caso, el programa y sus datos asociados se dividen en un conjunto de **segmentos**.
- **No es necesario que todos los segmentos** de todos los programas tengan la misma longitud, aunque existe una longitud máxima de segmento.

Segmentación Simple

- Como en la paginación, una dirección lógica segmentada consta de dos partes, en este caso un numero de segmento y un desplazamiento.

Segmentación Simple

- Como consecuencia del empleo de segmentos de distinto tamaño, la segmentación resulta similar a la partición dinámica.

Segmentación Simple

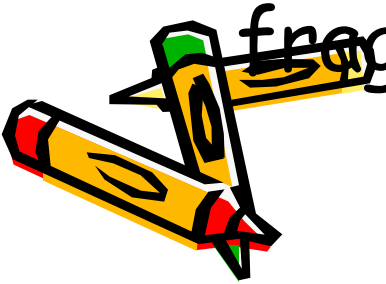
- En ausencia de un esquema de superposición o del uso de memoria virtual, seria necesario cargar en memoria todos los segmentos de un programa para su ejecución.

Segmentación Simple

- La diferencia, en comparación con la partición dinámica, radica en que, con segmentación, un programa puede ocupar mas de una partición y estas no tienen por que estar contiguas.

Segmentación Simple

- La segmentación elimina la fragmentación interna, pero, como la partición dinámica, sufre de fragmentación externa.
- Sin embargo, debido a que los procesos se dividen en un conjunto de partes mas pequeñas, la fragmentación externa será menor.

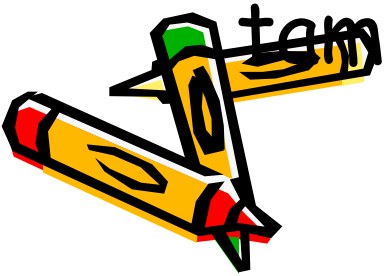


Segmentación Simple

- Mientras que la paginación es transparente al programador, la segmentación es generalmente visible y se proporciona como una comodidad para la organización de los programas y datos.
- Normalmente, el programador o el compilador asigna los programas y los datos a diferentes segmentos.

Segmentación Simple

- En aras de la programación modular, el programa o los datos pueden ser divididos de nuevo en diferentes segmentos.
- El principal inconveniente de este servicio es que el programador debe ser consciente de la limitación de tamaño máximo de los segmentos.



Segmentación Simple

- Otra consecuencia del tamaño desigual de los segmentos es que no hay una correspondencia simple entre las direcciones lógicas y las direcciones físicas.
- De forma análoga a la paginación, un esquema de segmentación simple hará uso de una tabla de segmentos para cada proceso y una lista de bloques libres en memoria principal.

Segmentación Simple

- Cada entrada de tabla de segmentos tendría que contener la dirección de comienzo del segmento correspondiente en memoria principal.

Segmentación Simple

- La entrada deberá proporcionar también la longitud del segmento para asegurar que no se usan direcciones no validas.
- Cuando un proceso pasa al estado Ejecutando, se carga la dirección de su tabla de segmentos en un registro especial del hardware de manejo de memoria.

Segmentación Simple

- En definitiva, con segmentación simple, un proceso se divide en varios segmentos que no tienen por que ser del mismo tamaño.
- Cuando un proceso se introduce en memoria, se cargan todos sus segmentos en regiones de memoria libres y se rellena la tabla de segmentos.

Memoria virtual o swap

- Las claves de este avance son dos características de la paginación y la segmentación.

swap

- 1. Todas las referencias a memoria dentro de un proceso son direcciones lógicas que se traducen dinámicamente a direcciones físicas durante la ejecución.

swap

- Esto quiere decir que un proceso puede cargarse y descargarse de memoria principal de forma que ocupe regiones diferentes en instantes diferentes a lo largo de su ejecución.

swap

- 2. Un proceso puede dividirse en varias partes (paginas o segmentos) y no es necesario que estas partes se encuentren contiguas en memoria principal durante la ejecución.

swap

- Esto es posible por la combinación de la traducción dinámica de direcciones en tiempo de ejecución y el uso de una tabla de paginas o de segmentos.

swap

- Volviendo sobre el avance comentado, si *estas dos características están presentes, no será necesario que todas las paginas o todos los segmentos de un proceso estén en memoria durante la ejecución.*

swap

- Si tanto la parte (pagina o segmento) que contiene la siguiente instrucción a leer como la parte que contiene los próximos datos a acceder están en memoria principal, la ejecución podrá continuar al menos por un tiempo.

swap

- El *termino fragmento* se refiere tanto a paginas como a segmentos, dependiendo de si se emplea paginación o segmentación.

swap

- Supóngase que se trae un proceso a memoria en un instante dado.
-

swap

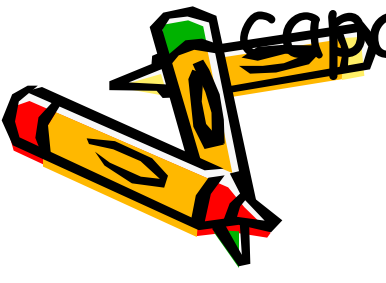
- El sistema operativo comienza trayendo solo unos pocos fragmentos, incluido el fragmento que contiene el comienzo del programa.

swap

- Se llamara **conjunto residente** del proceso a la parte de un proceso que está realmente en memoria principal.

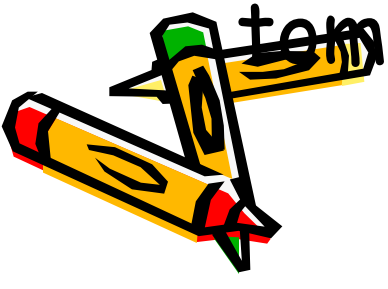
swap

- Cuando el proceso se ejecuta, todo estará bien mientras todas las referencias a memoria estén en posiciones que pertenezcan al conjunto residente.
- A través de la tabla de paginas o de segmentos, el procesador siempre es capaz de determinar si esto es así.



swap

- Si el procesador encuentra una dirección lógica que no está en memoria principal, genera una interrupción que indica un fallo de acceso a memoria.
- El sistema operativo pasa el proceso interrumpido al estado bloqueado y toma el control.



swap

- Para que la ejecución de este proceso siga más tarde, el sistema operativo necesita traer a memoria principal el fragmento del proceso que contiene la dirección lógica que provoco el fallo de acceso.

swap

- Para ello, el sistema operativo emite una solicitud de Lectura de E/S a disco.

swap

- Después de haber emitido la solicitud de E/S, el sistema operativo puede despachar otro proceso para que se ejecute mientras se realiza la operación de E/S.

swap

- Una vez que el fragmento deseado se ha traído a memoria principal y se ha emitido la interrupción de E/S, se devuelve el control al sistema operativo, que coloca el proceso afectado en el estado de Listo.

swap

- Ahora bien, esto puede llevar a cuestionarse la eficiencia de esta operación, en la cual un proceso puede estar siendo ejecutado y ser interrumpido sin otra razón que un fallo en la carga de todos los fragmentos necesarios para el proceso.

swap

- Las implicaciones de esta nueva estrategia, son dos, siendo la segunda más sorprendente que la primera.

swap

- Ambas conducen a mejoras en la utilidad del sistema:

swap

- 1. Se pueden conservar mas procesos en memoria principal.
- Puesto que se van a cargar solo algunos fragmentos de un proceso particular, habrá sitio para mas procesos.

swap

- Esto conduce a una utilización más eficiente del procesador, puesto que es más probable que, por lo menos, uno de los numerosos procesos este en estado Listo en un instante determinado.

swap

- 2. Es posible que un proceso sea más grande que toda la memoria principal.
- Se elimina así una de las limitaciones más notorias de la programación.
- Sin el esquema que se ha expuesto, un programador debe ser consciente de cuanta memoria tiene disponible.

swap

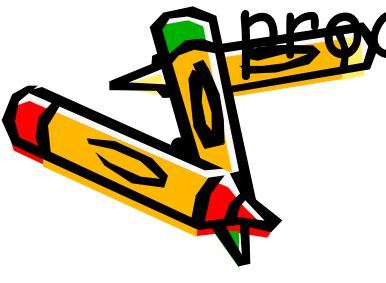
- Si el programa que esta escribiendo es demasiado grande, el programador debe idear formas de estructurar el programa en fragmentos que puedan cargarse de forma separada con algún tipo de estrategia de superposición.

swap

- Con una memoria virtual basada en paginación o segmentación, este trabajo queda para el sistema operativo y el hardware.

swap

- En lo que atañe al programador, se las arregla con una memoria enorme, dependiendo del tamaño de almacenamiento en disco.
- El sistema operativo cargara automáticamente en memoria principal los fragmentos de un proceso cuando los necesita.



swap

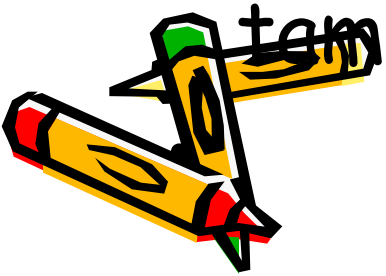
- Como los procesos se ejecutan solo en memoria principal, a esta memoria se le llama **memoria real**.
- Pero un programador o usuario percibe en potencia una memoria mucho mayor, que está situada en el disco.

swap

- Esta última se conoce por **memoria virtual. O swap**
- La memoria virtual permite una multiprogramación muy efectiva y releva al usuario de las rígidas e innecesarias restricciones de la memoria principal.

swap

- Para un aprovechamiento eficiente del procesador y de los servicios de E/S es conveniente mantener tantos procesos en memoria principal como sea posible.
- Además, conviene liberar a los programadores de las limitaciones de tamaño en el desarrollo de programas.



swap

- La forma de abordar ambos problemas es por medio de la memoria virtual.
- Con memoria virtual, todas las referencias a direcciones son referencias lógicas que se traducen a direcciones reales durante la ejecución.

swap

- Esto permite a los procesos situarse en cualquier posición de memoria principal y cambiar de ubicación a lo largo del tiempo.

swap

- La memoria virtual permite también dividir un proceso en fragmentos.

swap

- Estos fragmentos no tienen por qué estar situados de forma contigua en la memoria principal durante la ejecución y no es ni siquiera necesario que todos los fragmentos del proceso estén en memoria durante la ejecución.

swap

- Los dos enfoques básicos de memoria virtual son la paginación y la segmentación.
- Con paginación, cada proceso se divide en páginas de tamaño fijo y relativamente pequeño.

swap

- La segmentación permite el uso de fragmentos de tamaño variable.
- También es posible combinar segmentación y paginación en un único esquema de gestión de memoria.
- Un esquema de gestión de memoria virtual exige un soporte tanto de hardware como de software.

swap

- El soporte de hardware lo proporciona el procesador.

swap

- Este soporte incluye la traducción dinámica de direcciones virtuales a direcciones físicas y la generación de interrupciones cuando una página o segmento referenciado no están en memoria principal.

swap

- Estas interrupciones activan el software de gestión de memoria del sistema operativo.
- Una serie de cuestiones de diseño relativas a los sistemas operativos dan soporte a la gestión de memoria virtual:

swap

- *Políticas de lectura:* Las páginas de los procesos pueden cargarse por demanda o se puede usar una política de paginación previa; esta ultima agrupa las actividades de entrada cargando varias páginas a la vez.

swap

- *Políticas de ubicación*: En un sistema de segmentación pura, un segmento entrante debe encajar en un espacio de memoria disponible.

swap

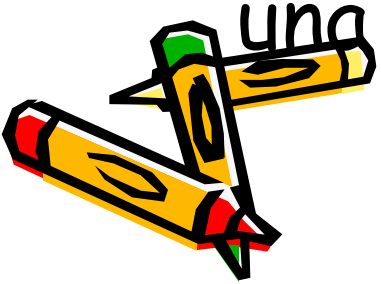
- *Políticas de reemplazo:* Cuando la memoria está llena, debe tomarse la decisión de que pagina o paginas serán reemplazadas.

swap

- *Administración del conjunto residente:* El sistema operativo debe decidir cuanta memoria principal ha de asignar a un proceso en particular cuando se carga.
- Puede hacerse una asignación estática en el momento de la creación del proceso o bien puede cambiar dinámicamente.

swap

- *Políticas de vaciado:* Las paginas modificadas de un proceso pueden escribirse al disco en el momento del reemplazo o bien puede aplicarse una política de paginación previa; esta ultima agrupa las actividades de salida escribiendo varias páginas de una vez.



swap

- *Control de carga:* El control de carga determina el numero de procesos residentes que habrá en memoria principal en un momento dado.