Memoria Virtual

Eloy Anguiano

Rosa Mª Carro

Ana González

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Parte I

Memoria virtual

Introducción
Conceptos

- Concepto de **Memoria Virtual**: Método para conseguir que la suma de los espacios de pila, datos y texto (código) de un programa pueda ser mayor que el tamaño físico de la memoria disponible para él. (Fotheringham, 1961)
- A cada proceso se le asigna un área de direcciones, pudiendo ser no contiguo.
- El SO mantiene en memoria solamente las partes del programa que se están utilizando y mantiene en disco (intercambiadas) el resto.
  - **Conjunto residente**: parte del proceso que está en la memoria principal (memoria real).
  - Un usuario percibe en potencia una memoria mucho mayor disponible para sus procesos, que combina la memoria principal con la parte del disco destinada a este fin. La memoria en total que parece que tiene disponible se llama (**Memoria Virtual**)
- Sirve para **sistemas de mono y multiprogramación**.
- Se necesita disponer del *hardware* que dé soporte a la gestión de la memoria virtual.

Introducción
Conceptos

Ventajas

- Permite **optimizar el uso de la memoria principal**:
  - Mantiene **'más procesos'** en memoria principal, albergando solo una parte (suficiente) de cada uno de ellos.
    - Permite la multiprogramación de forma efectiva
  - Mantiene en disco partes del proceso poco usadas (rutinas de atención a errores poco frecuentes, funciones de uso esporádico, datos menos usados, ...)
- Permite que un proceso sea más grande que toda la memoria principal.
- El encargado de gestionar la memoria virtual es el sistema operativo.

Introducción
Conceptos

Problemas

- Fallos de direccionamiento: se intenta acceder a una posición/dirección (de una instrucción/dato) que no está en memoria principal
  1. Se genera una interrupción, indicando fallo de acceso a memoria
  2. El proceso pasa a **estado bloqueado**, y el SO a ejecución
  3. El SO emite una solicitud de E/S al disco
  4. El SO expide otro proceso para que se ejecute (**planificador del S.O.**)
  5. La operación de E/S se realiza, actualizándose la memoria principal. Llega interrupción indicando que la operación de E/S se ha realizado.
  6. Se devuelve el control al S.O., que pasa el proceso a **estado listo**. El planificador decidirá cuándo debe pasar a ejecución.
- En algunos casos, se puede dar hiperpaginación (hyperthrashing)
  - El SO más tiempo intercambiando bloques que ejecutando procesos.

Parte II

Paginación

Conceptos

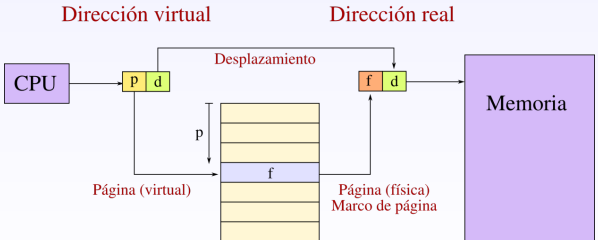
- La **memoria física** se divide en bloques de tamaño fijo: **marcos**.
- La **memoria virtual** se divide en bloques del mismo tamaño:**páginas** (los procesos se dividen en páginas).
- Al ejecutar un proceso (también para ponerlo listo), se cargan sus páginas en los marcos disponibles. La vinculación de direcciones requiere soporte por **hardware (Manejador de Memoria)**.
- En un sistema con paginación **no se produce fragmentación externa y sí fragmentación interna**.
- Tabla de páginas:
  - Normalmente una por proceso (más adelante veremos que en procesos grandes se complica)
  - Requiere **soporte por hardware**: manejador de memoria (MMU)
- Un intento de acceso a una página virtual que no esté asociada a un marco produce un señalamiento al SO (trap), llamado **fallo de página**.

Conceptos

Acciones de respuesta del SO en un fallo de página

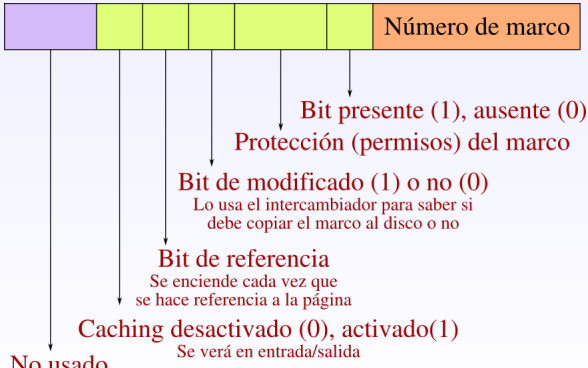
- Si no hay marcos disponibles:
  - Se selecciona un marco asociado a una página poco usada del proceso.
  - Se intercambia la página a disco.
  - Asigna el marco de la página liberada a la página virtual que se intenta acceder.

Reasignación dinámica por bloques de las direcciones de memoria del proceso



Tablas de páginas

- Una tabla de páginas por proceso, con tantas entradas como páginas virtuales tiene el proceso.
  - Cada entrada contiene el número de marco en que está cargada la página en memoria principal.
- Estructura de una entrada de la Tabla de Procesos:



Tablas de páginas

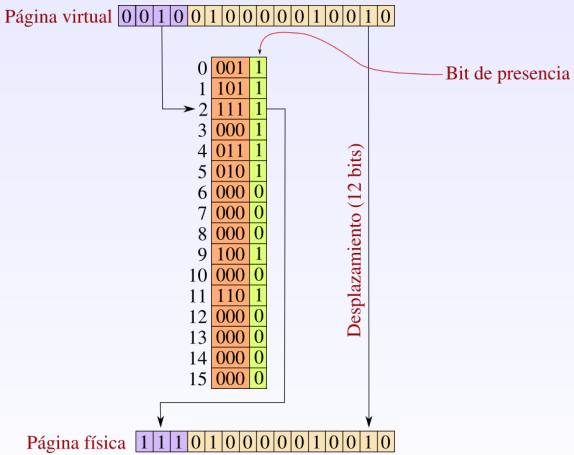
- Contiene (para cada proceso) el número de marco que corresponde a cada página virtual del proceso.
- Normalmente **una tabla de páginas por proceso**
- Las tablas de páginas de los procesos tienen longitud variable (dependen del tamaño de cada proceso - 1 entrada por cada página).
- La tabla está **cargada en memoria principal** y, si es muy grande, puede estar sujeta a su vez a paginación $\Rightarrow$ Paginación multinivel

Al ejecutar un proceso ...

1. La dirección de comienzo de la tabla de páginas de ese proceso se mantiene en un registro
2. Se cargan las páginas necesarias en los marcos disponibles

Tablas de páginas

- El tamaño de página viene definido por el hardware y suele ser una potencia de 2 que varía entre 512 Bytes y 16 MB.
- Ej. simplificado: Direccionamiento de un espacio virtual de 64 kB, distribuido en 16 páginas de 4 kB.



Tablas de páginas
Aspectos de diseño

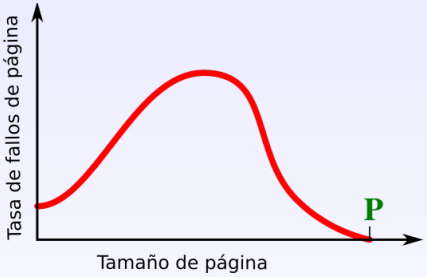
- El **tiempo de asociación** (página-marco) debe ser reducido
  - Soluciones basadas completamente en hardware (utilizando registros) son las más rápidas, pero esto sólo es válido si las tablas de páginas son pequeñas.
- Relativo al **tamaño de las páginas**:
  - Cuanto menor sea el tamaño de página, menor será la cantidad de fragmentación interna (En promedio la mitad del tamaño de la página).
  - Cuanto menor sea la página, mayor será el número de páginas que se necesitan por proceso.
  - Un número mayor de páginas por proceso significa que las tablas de páginas serán mayores.
  - Esto puede significar que una gran parte de las tablas de páginas de los procesos activos deben estar en la memoria virtual.

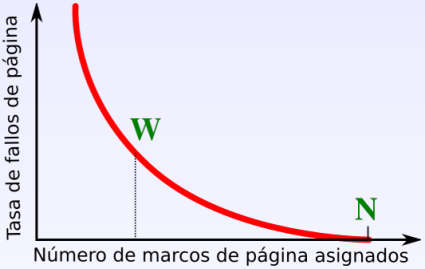
Tablas de páginas
Aspectos de diseño

- Considerando páginas pequeñas:
  - Si el tamaño de página es muy pequeño, estarán disponibles en la memoria principal un gran número de páginas para cada proceso.
  - Después de un tiempo, todas las páginas de la memoria contendrán parte de las referencias más recientes del proceso. La tasa de fallos de página será menor.
- Considerando páginas grandes:
  - La memoria secundaria está diseñada para transferir eficazmente los bloques de datos de mayor tamaño, de manera que es propicia para tamaños de página mayores.
  - Cuando se incrementa el tamaño de la página, cada página individual contendrán posiciones cada vez más distantes de cualquier referencia reciente. La **tasa de fallos** será mayor.

Tablas de páginas
Aspectos de diseño

El comportamiento típico de la paginación en un programa es el siguiente:





Donde:

- **P**: es el tamaño del proceso completo.
- **W**: es el tamaño del conjunto de trabajo.
- **N**: es el número total de páginas del proceso.

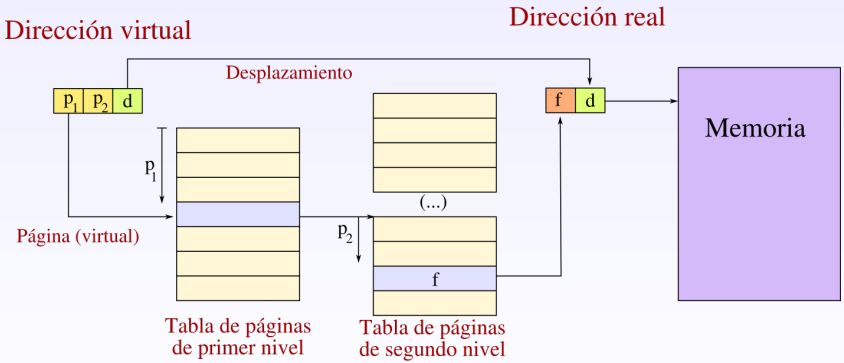
Tablas de páginas
Aspectos de diseño

Soluciones al tamaño de tablas grandes

- Paginación multinivel
  - Una parte de las tablas de páginas deben estar en la memoria virtual, sujetas a paginación también.
- Utilizar tablas de páginas invertidas en lugar de tablas de páginas.

Tablas de páginas
Paginación Multinivel

- **Objetivo**: evitar tener siempre en memoria tablas de páginas completas.
- **Solución**: dividir la tabla en sub-tablas y mantener en memoria sólo las que sean necesarias en cada momento.



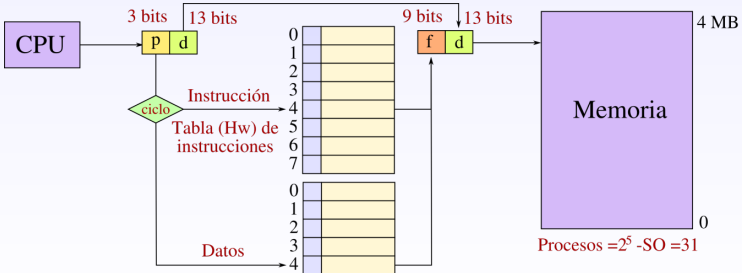
Ejemplos de tamaños de página en función del Hardware

Computadora	**Tamaño de página**
Atlas	512 palabras de 48 bits
Honeywell-Multics	1.024 palabras de 36 bits
IBM 370/XA y 370/ESA	4 Kbytes
Familia VAX	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	de 4 Kbytes a 16 Mbytes
UltraSPARC	de 8 Kbytes a 4 Mbytes
Pentium	de 4 Kbytes a 4 Mbytes
Power Pc	4 Kbytes

Tablas de páginas

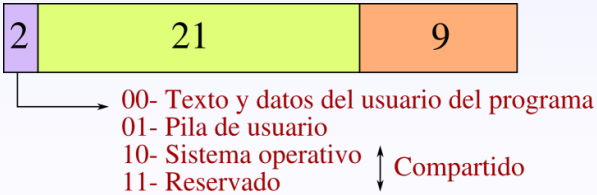
De un nivel, con 2 tablas/proceso: Ejemplo DEC PDP–11

- Direccionamiento con 16 bits a nivel de byte $\rightarrow 2^{16}B$=64KB
- Tamaño de página: 8 KB (2^{13}B) $\rightarrow$ 13 bits para indicar desplazamiento.
- Memoria física: 4 MB (2^{22}B). Como cada marco ocupa 8KB (=página=2^{13}B) $\rightarrow$ Hay 2^9=512 marcos $\rightarrow$ 9 bits para indicar nº marco.
- Mantiene 2 tablas separadas: instrucciones y datos. 16-13=3 bits para indexar nº página en cada tabla. Memoria lógica del proceso=$2x2^{16}$B=2^7KB =128 KB.



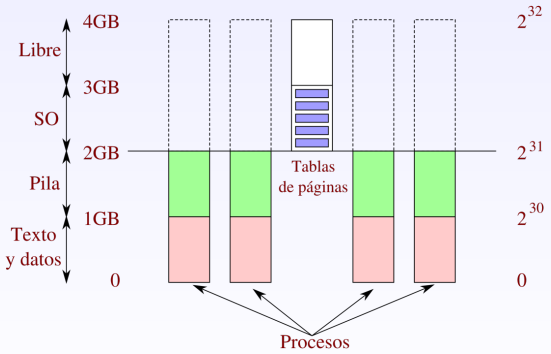
Tablas de páginas
De dos niveles: Ejemplo VAX

- Direccionamiento a nivel de byte con 32 bits→ memoria lógica de 2^{32}=4GB
- Tamaño de página: 512 bytes (2^9B).
- Espacio de direcciones lógico divido en cuatro secciones, cada una de 1GB (2^{30} bytes).
- Entrada de la tabla de página: 4 bytes.
- Tamaño de la tabla de páginas ($2^{21} \times 4B$): ¡¡8 MB!!
- Memoria física $<$ 2 MB (en VAX más pequeñas). ¡No cabe la tabla!
- Direcciones lógicas:



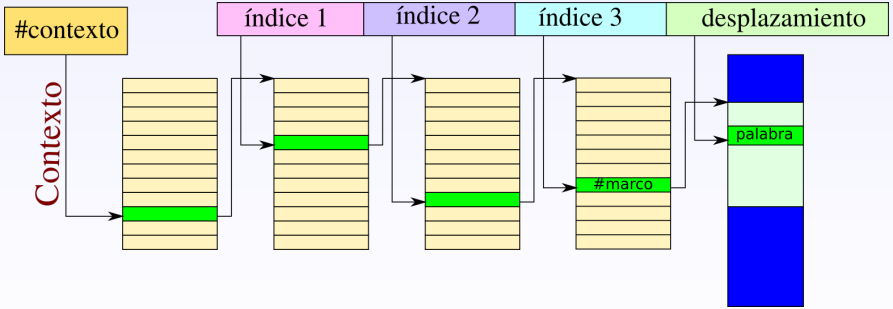
Tablas de páginas
De dos niveles: Ejemplo VAX

- Solución: las tablas de páginas están también paginadas, y las partes no usadas de la tabla se sitúan en el disco.
- Sólo el SO y su tabla de páginas permanece **permanentemente** en memoria
- Esquema de memoria virtual de los procesos:



Tablas de páginas
De tres niveles: Ejemplo SPARC

- Direccionamiento con 32 bits (memoria lógica de 4 GB $=2^{32}$B)
- Tamaño de página: 4KB (2^{12}B) $\Rightarrow 2^{20}$ páginas (algo más de un millón de páginas).
- Se define un **contexto**, único para cada proceso, y una **tabla de contexto** (hardware) para almacenar un apuntador al comienzo de la tabla de alto nivel del proceso.



Tablas de páginas
Rendimiento de un sistema de paginación

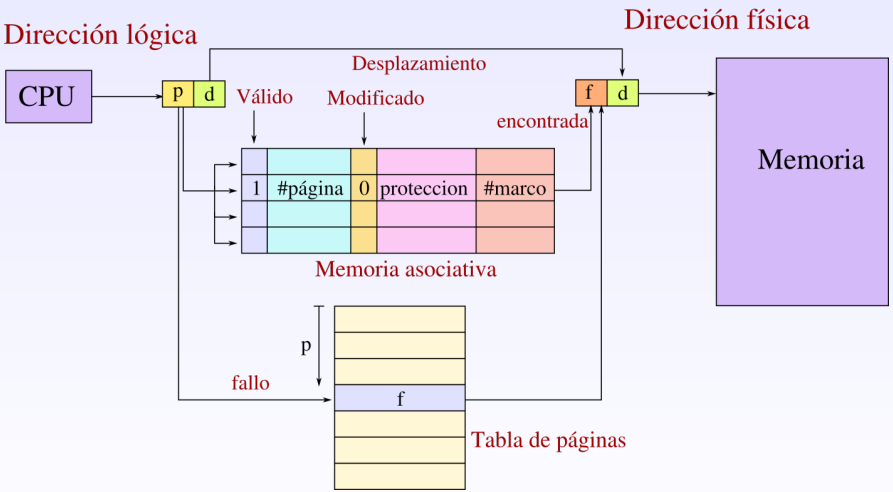
- El tiempo de acceso efectivo a memoria (t_{ae}) para un sistema de paginación de memoria mononivel es $t_{ae} = t_b + (1 - p) \times t_{am} + p \times t_{fallo} + t_{am}$
  donde:
  t_b: tiempo medio de búsqueda en la tabla de páginas
  p: probabilidad de que ocurra un fallo de página
  t_{am}: tiempo de acceso a memoria
  t_{fallo}: tiempo de resolución de un fallo de página. Tiene tres componentes principales:
  1. Atender la interrupción de fallo de página
  2. Traer la página del disco a la memoria principal
  3. Continuar con el proceso
- En el caso de tablas multinivel (n niveles):
  $t_{ae} = \sum_{i=1}^{N} [t_{bi} + (1 - p_i) \times t_{am} + p_i \times t_{fallo}] + t_{am}$

Memoria Asociativa
Conceptos

- **Translation Lookaside Buffer (TLB o Buffer de Traducción Adelantada)**: solución para acelerar la traducción de direcciones y, por tanto, el acceso a los marcos, cuando las tablas de procesos son muy grandes (búsqueda lenta) y/o están organizadas en niveles (requiere múltiples accesos a memoria).
- **Observación**: los procesos acceden normalmente a un número pequeño de páginas (y esporádicamente al resto).
- **Mejora/solución**: dotar a los ordenadores con Hw (memoria asociativa, caché de memoria administrada por la MMU) para asociar algunos nº páginas de uso frecuente con los nº marcos en que están en MP sin necesidad de acceder a la tabla de páginas. Tamaño de la memoria asociativa usual: de 8 a 32 entradas.
- **Proporción de encuentros**: Proporción de accesos a la Memoria Asociativa que son exitosos (la Página Virtual buscada se encuentra en la Memoria).

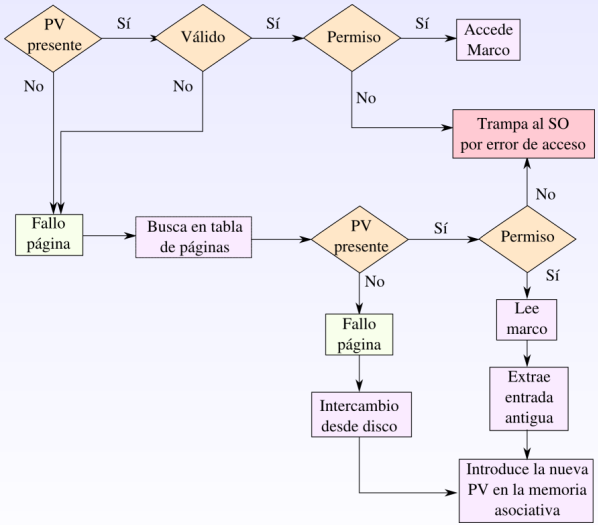
Memoria Asociativa

Esquema



Memoria Asociativa

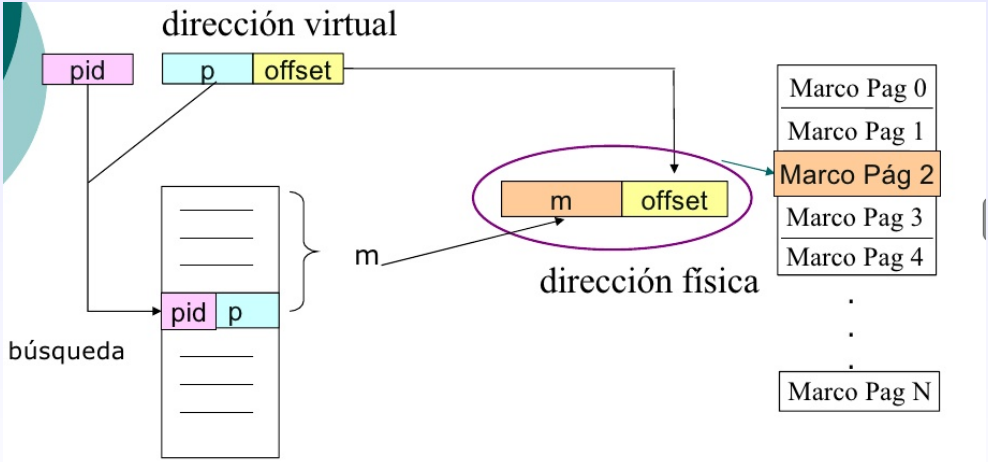
Asignación del manejador de memoria



Tablas de páginas invertidas
Conceptos

- Sistemas con direccionamiento por 64 bits a nivel de Byte (16 EB= 2^{64}B), el número de páginas de tamaño (p.ej) 4 KB es 2^{52} $\Rightarrow$ Impensable mantener tablas de páginas de esa longitud, mayor que la memoria física de la mayoría de los sistemas.
- En sistemas donde el número de marcos físicos es sustancialmente menor, permite organizar la tabla de entradas alrededor de la memoria física en lugar de la memoria virtual (IBM S/38, HP Spectrum).
- La tabla de páginas invertida tiene **tantas entradas como marcos**, y cada entrada contiene la dirección virtual de la página.
- Se utiliza siempre con una **memoria asociativa**.
- Si una dirección virtual no se encuentra en la tabla invertida, se hace una búsqueda en una tabla convencional, que puede estar en memoria o en disco.
- Si se encuentra, se obtiene el número de marco.

Tablas de páginas invertidas
Conceptos



Una tabla de página en el SO para todos los procesos

Parte III

Políticas del gestor de memoria

Características

Encaminadas a minimizar porcentaje de fallos de páginas para maximizar el rendimiento

¿De qué depende el rendimiento?

- Del tamaño de memoria principal
- De la velocidad relativa de memoria principal y secundaria
- Del tamaño y número de procesos que compiten por los recursos
- Del comportamiento de programas individuales:
  - Aplicación
  - Lenguaje de programación y compilador
  - Estilo de programador
  - Comportamiento dinámico del usuario en programas interactivos

Tipos de políticas

No hay una política definitivamente mejor que las otras

- Políticas de Lectura
  - ¿Cuándo cargo una página en memoria principal?
- Políticas de Ubicación
  - ¿Dónde coloco la nueva página?
- Políticas de Reemplazo
  - Si la memoria principal está llena ¿qué página/s reemplazo?
- Gestión del Conjunto Residente
  - ¿Cuánta memoria principal debe asignarse a un proceso cuando se carga?
- Políticas de Vaciado
  - ¿Cuándo escribo una página modificada en memoria secundaria?
- Control de Carga
  - ¿Cuál es el grado de multiprogramación?

Tipos de políticas
Políticas de Lectura (fetch)

¿Cuándo cargar una página en memoria principal?

- **Paginación por Demanda**: Se carga una página sólo cuando se produce un fallo en esa página
  - Número elevado de fallos al inicio
  - Con el tiempo y si no hay cambio de localización:
    - Estabilización gracias al principio de cercanía.
    - El número de fallos disminuye hasta un nivel bajo.
- **Paginación Previa**: se carga la página que ha producido el fallo y las páginas cercanas a ésta.
  - Aprovecha características de los discos

Tipos de políticas
Políticas de Ubicación

¿Dónde va a residir una parte de un proceso en la memoria principal?

- Es una decisión importante en sistemas con segmentación
  - Mejor ajuste, primer ajuste, siguiente ajuste, etc
- Carece de importancia en sistemas con paginación (con y sin segmentación)

Tipos de políticas
Políticas de Reemplazo

Si la memoria está ocupada: ¿Qué páginas se eligen para ser reemplazadas?

Gestión del Conjunto Residente

- ¿Número de marcos a asignar para cada proceso?
- ¿Reemplazar sólo páginas del propio proceso o de cualquier otro?

Algoritmos de Reemplazo

- Tras un fallo de página y con toda la memoria principal ocupada, se debe elegir qué página situada en memoria se lleva al disco para dejar el marco libre.
- **Objetivo**: reemplazar la página con menor posibilidad de ser referenciada en un futuro cercano $\Rightarrow$ Intentar predecir el comportamiento futuro en función del comportamiento pasado.
- Cuanto más elaborada es la política, mayor sobrecarga.

Tipos de políticas

Políticas de gestión del conjunto residente

Tamaño del Conjunto Residente

El S.O. decide cuánta memoria asignar a un proceso (nº marcos = nº páginas que puede cargar en memoria principal)

Factores positivos que influyen en la decisión

Cuanto menos memoria necesite cada proceso, mayor cantidad de procesos en memoria $\Rightarrow$ mayor probabilidad de procesos en estado listo $\Rightarrow$ aumenta el grado de multiprogramación.

Factores negativos que influyen en la decisión

- Si hay pocas páginas de un proceso en memoria, aumenta la probabilidad de fallos de página.
- Por encima de un determinado número, más memoria no tendrá un efecto notable (por el principio de cercanía: se tendrán cargadas las páginas que se necesitan durante algún tiempo).

Tipos de políticas

Políticas de gestión del conjunto residente

Asignación Fija: otorga a cada proceso un número fijo de páginas

- Se decide la cantidad de memoria asignada al proceso en la carga inicial, según el tipo de proceso o directrices del programador o administrador.
- Cuando hay fallos de página, siempre se reemplaza una página del mismo proceso.

Problemas

- **Asignación por exceso**:
  - Se desperdicia espacio: el proceso no necesita todos los marcos que se le han asignado.
  - Posible procesador ocioso: hay pocos procesos en memoria principal.
- **Asignación por defecto**:
  - Alto porcentaje de fallos de páginas, aunque en el sistema haya marcos vacíos
  - Multiprogramación más lenta (intercambios páginas con disco).

Tipos de políticas
Políticas de gestión del conjunto residente

Asignación Variable: el número de marcos de un proceso cambia durante su vida

- Cambia en función de la tasa de fallos de página. Se asignará dinámicamente el nº de marcos atendiendo a las necesidades del proceso.
- La asignación variable es más potente pero costosa (el S.O. debe evaluar el comportamiento de los procesos dinámicamente).

Tipos de políticas
Políticas de gestión del conjunto residente

Los **algoritmos de reemplazo de páginas** pueden ser de:

- **Ámbito global**. Un proceso puede utilizar marcos que pertenecen a otro proceso (ej: en asignación con prioridades).
  - Sólo tiene sentido con asignación variable.
- **Ámbito local**. Un proceso sólo puede utilizar marcos que le han sido asignados a dicho proceso.
  - Tiene sentido en asignación fija y en asignación variable.
  - **Problemas:**
    - Desperdicia espacio si el proceso no necesita todos los marcos que se le han asignado.
    - Si el proceso necesita más marcos de los asignados, no podrá ejecutarse correctamente, aunque en el sistema haya muchos marcos vacíos.

Tipos de políticas

Políticas de gestión del conjunto residente

Asignación fija y reemplazo de ámbito local

Al cargar un nuevo proceso, se le asigna un número de marcos fijo (depende de aplicación y solicitud del programa)

Desventajas

- Si se asignan pocas páginas se produce un alto porcentaje de fallos de páginas.
- Si se asignan muchas páginas hay pocos procesos en memoria $\Rightarrow$ procesador ocioso

Tipos de políticas

Políticas de gestión del conjunto residente

Asignación variable y reemplazo de ámbito local

- Al cargar un nuevo proceso, se le asigna un número de marcos (paginación previa o bajo demanda)
- Cuando hay un fallo de página con reemplazo se selecciona una página del proceso que produjo el fallo de página
- De vez en cuando, se evalúa la asignación de un proceso y se aumenta o disminuye para mejorar el rendimiento global.

Tipos de políticas

Políticas de gestión del conjunto residente

Asignación variable y reemplazo de ámbito global

Sencilla de implementar

Fallos de página: Nuevo marco libre para el proceso, donde se carga la página. Casi siempre se incrementa la memoria asignada al proceso.

Dificultad: La elección del reemplazo se realiza entre todos los marcos.

Estrategia: Uso de memoria intermedia para las páginas reemplazadas. Mejora el tiempo de recuperación de páginas recientemente reemplazadas (están en memoria).

Cuestiones a resolver

- Tamaño del conjunto residente.
- Periodicidad de los cambios del conjunto residente.

Solución

Estrategia del **Conjunto de trabajo**

Tipos de políticas
Conjunto de Trabajo

- Se utiliza para determinar el tamaño del conjunto residente y el momento de los cambios
- El conjunto de trabajo de un proceso en un instante t es $W(t, \Delta t)$: conjunto de páginas a las que el proceso ha hecho referencia en las últimas Δt unidades de tiempo
- $W(t, \Delta t)$:
  - W crece con Δt
  - W depende del instante t: puede ser 1 o llegar hasta el número total de páginas del proceso
  - Su tamaño puede variar durante la ejecución
  - Crece al iniciarse un proceso
  - Estabilidad (principio de cercanía)
  - Las oscilaciones indican el desplazamiento del programa a otra ubicación

Tipos de políticas
Conjunto de Trabajo

- Si el número de marcos disponibles es inferior al tamaño del conjunto de trabajo, se producirán frecuentes fallos de página
- Un proceso hiperpaginado pasa más tiempo intercambiando páginas que ejecutándose, y puede "robar" páginas de otros procesos, provocando, a su vez, su hiperpaginación (hypertrashing o trasiego)
- Consecuencia del hypertrashing: reducción drástica del uso de CPU $\Rightarrow$ Aumenta el número de procesos en estado bloqueado.
- La hiperpaginación se limita con asignación local, y si se asigna a cada proceso un número de marcos suficiente.

Problemas

- ¿Cómo calcular el número de marcos que un proceso necesita?
- ¿Cómo calcular tamaño del conjunto de trabajo para un proceso?

Tipos de políticas
Modelo del conjunto de trabajo

Finalidad: reducir la tasa de fallos de página

Principio de localidad/cercanía: las páginas que se utilizan en cada momento se pueden agrupar en grupos que varían a lo largo de la ejecución del proceso (P.e., una llamada a una función provoca un cambio de localidad).

Tipos de políticas
Modelo del conjunto de trabajo

Estrategia

- **Supervisar** el conjunto de trabajo de cada proceso
- **Eliminar periódicamente** del conjunto residente las páginas que no pertenezcan a su conjunto de trabajo
- Un proceso **se puede ejecutar sólo si** su conjunto de trabajo esta en memoria principal (en conjunto residente)

Problemas

- El pasado no siempre predice el futuro
- Es impracticable una medida real del conjunto de trabajo de cada proceso
- El valor óptimo de Δt a considerar es desconocido y puede variar

Tipos de políticas
Modelo del conjunto de trabajo

Estrategia alternativa

Algoritmo de frecuencia de fallos de página

- Cada página tiene un bit de uso que se pone a 1 cuando accede a la página
- Al producirse fallo de página el S.O. mira el tiempo transcurrido desde el último fallo de página para el proceso
  - Si el tiempo es menor que F (umbral), se añade una página al conjunto residente.
  - En caso contrario:
    - Se descartan las páginas con el bit de uso a 0, reduciéndose el conjunto residente.
    - Se restaura a 0 el valor del bit de uso en las páginas restantes

Estrategia alternativa (Cont.)

Problema: periodos de transición entre dos regiones de referencia

El conjunto residente crece antes de que las páginas de la antigua región de referencia hayan sido expulsadas y provoca:

- Aumento del grado de multiprogramación, sobrecargando la CPU.
- Suspensión de procesos innecesaria

Tipos de políticas
Modelo del conjunto de trabajo

Otra estrategia alternativa

Política de conjunto de trabajo con muestreos en intervalos variables:

Evalúa el conjunto de trabajo mediante muestras temporales:

- Al comienzo de un intervalo de muestreo, se restauran los bits de uso de todas las páginas residentes del proceso
- Al final, sólo las páginas que han sido referenciadas se mantendrán en el conjunto residente para el próximo intervalo de tiempo

Tipos de políticas
Políticas de Vaciado

Vaciado por demanda

Vaciado por demanda: Se escribe la página en disco **sólo cuando se va a reemplazar** (antes de leer la siguiente)

Ventaja: Minimiza escrituras en disco

Desventaja: Cada fallo de página produce dos transferencias de página (menor rendimiento)

Vaciado previo

Escribe las páginas modificadas **antes de que se necesiten sus marcos**, escribiéndolas por lotes

Ventaja: Aprovecha las ventajas de escribir en el disco por lotes

Desventaja: Posibles operaciones innecesarias: escritura de páginas que van a ser modificadas antes de ser reemplazadas

Tipos de políticas
Políticas de Vaciado

Alternativa: incorporar almacenamiento intermedio

- Mantiene temporalmente las páginas recien reemplazadas en una zona de la memoria reservada para ello (simil: contenedores de basura)
- Dos tipos de contenedores:
  - Lista de páginas **modificadas**
  - Lista de páginas **no modificadas**
- Lista de modificadas son enviadas a disco periódicamente por **lotes** y después se realiza un intercambio de los contenedores:
  - Conjunto de páginas modificadas se convierte en el contenedor de no modificadas. (Veremos por qué en el tema de E/S).
  - Conjunto de páginas no modificas pasa a ser el de modificadas.

Tipos de políticas
Control de carga

Grado de multiprogramación

- Si hay pocos procesos es probable que todos estén en estado bloqueado
- Si hay muchos, el tamaño medio del conjunto de trabajo no es el adecuado: aumenta la frecuencia de fallos de página (posible hiperpaginación)

Grado de multiprogramación se debe aumentar cuando el número de fallos de página sea pequeño y disminuir cuando sea grande

Tipos de políticas
Control de carga

Criterios de suspensión de procesos

Baja prioridad: Decisión de política de planificación

Muchos fallos de página: Conjunto de trabajo inadecuado

Último activado: Conjunto de trabajo residente no formado

Conjunto residente más pequeño: Penaliza procesos con ubicaciones pequeñas

Proceso mayor: Libera una cantidad de marcos grande

Mayor ventana de ejecución restante: Planificación: primero el proceso más corto

La elección depende de los objetivos y el tipo de programa

Parte IV

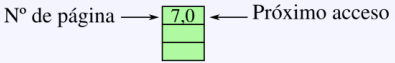
Algoritmos de reemplazo

Conceptos

- Tras un fallo de página, el SO debe elegir qué marco de página de la memoria deber ser intercambiado a disco para hacer sitio a la nueva página que se está solicitando.
- Criterio general: sustituir marcos de páginas poco usados por el proceso.

Óptimo

- Cada página contiene una etiqueta con el número de instrucciones que transcurrirán hasta el próximo acceso a la página. Se reemplaza la página que tenga la etiqueta más alta.
- **Problema**: Es **irrealizable** ya que el SO no tiene forma de saber cuándo se va a realizar un nuevo acceso a una página. Es posible si se ejecuta en modo simulación, se computan los accesos a las páginas, y esos cómputos se utilizan en sucesivas ejecuciones.



Tiempo	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Página	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0
Marcos	7,x	7,x 0,3	7,x 0,2 1,B	2,5 0,1 1,A	2,4 0,2 1,9	2,3 0,1 3,4	2,2 0,4 3,3	2,1 4,x 3,2	2,4 4,x 3,1	2,3 4,x 3,2	2,2 0,5 3,1	2,1 0,4 3,x	2,2 0,3 3,x	2,1 0,2 1,x	2,x 0,1 1,x	2,x 0,x 1,x

LRU (Least Recently Used)

- Implementa una aproximación del algoritmo óptimo, basado en mirar al pasado y a partir de él estimar cuál podría ser el uso de la página.
- Selecciona el marco en el que está la página que lleva más tiempo sin acceso (la que hace más tiempo que no se referencia).
- Hay varias implementaciones posibles.

LRU (Least Recently Used)

Posibles implementaciones

La tabla se implementa como pila

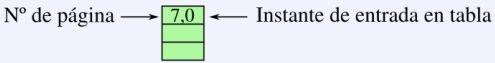
- Pila de números de página que conserva en la salida la página más recientemente usada y en la base la menos recientemente usada
- La actualización de la tabla es muy lenta, aún usando HW especial.

Uso de contadores

- Las entradas de la tabla de páginas tienen un campo de 'tiempo de uso' en el que el Manejador de Memoria escribe el tiempo de cada referencia.
- Cada acceso a memoria requiere una búsqueda en la Tabla de Páginas y una escritura en memoria por cada acceso a memoria.
- Requiere HW especial.

FIFO

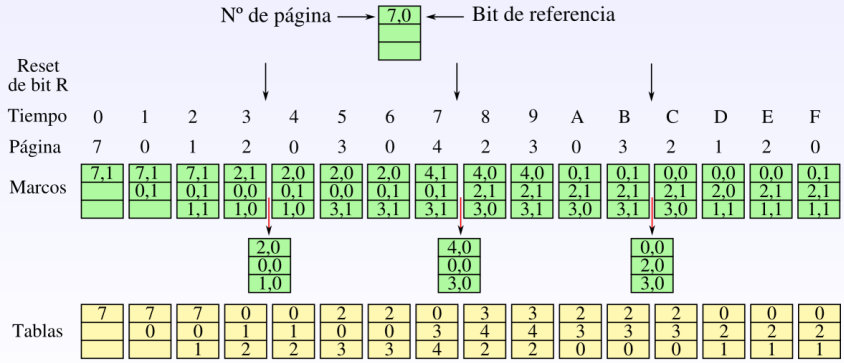
- Cada entrada de la tabla de páginas tiene un registro asociado con el instante de carga de la página, o (b) el SO mantiene la tabla de páginas en orden de antigüedad (FIFO)
- Cuando se produce un fallo de página y no hay marcos libres, se intercambia a disco la página que lleve más tiempo en la tabla.



Tiempo	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Página	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0
Marcos	7,0	7,0 0,1	7,0 0,1 1,2	2,3 0,1 1,2	2,3 0,1 1,2	2,3 3,5 1,2	2,3 3,5 0,6	4,7 3,5 0,6	4,7 2,8 0,6	4,7 2,8 3,9	0,A 2,8 3,9	0,A 2,8 3,9	0,A 2,8 3,9	0,A 1,D 3,9	0,A 1,D 2,E	0,A 1,D 2,E
Tablas	7	7 0	7 0 1	0 1 2	0 1 2	1 2 3	2 3 0	3 0 4	0 4 2	4 2 3	2 3 0	2 3 0	2 3 0	3 0 1	0 1 2	0 1 2

Segunda oportunidad

- Modificación de FIFO. Se revisa el bit de uso o referencia de la entrada más antigua: si es 1, se pone a 0 y se sitúa al final de la cola de páginas. Así sucesivamente hasta que se encuentre una página no referenciada.
- Posibilidad de combinarlo con resets del bit de uso (U) cada cierto tiempo.

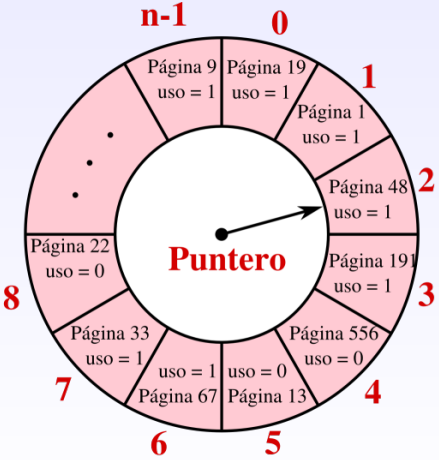


Del reloj

- Las páginas se mantienen en una cola circular, con un puntero a la página más antigua (cargada hace más tiempo en memoria). Si hay un fallo de página y el bit de uso del marco apuntado es U=0, la página que estaba en ese marco se reemplaza y se avanza el puntero. Si U=1, se pone U=0 y se avanza hasta encontrar una página con U=0 (página que se reemplaza, y el puntero avanza al siguiente marco).
- Es solo una implementación alternativa del algoritmo de segunda oportunidad, más eficiente: es el puntero el que se desplaza, no las entradas de la tabla.

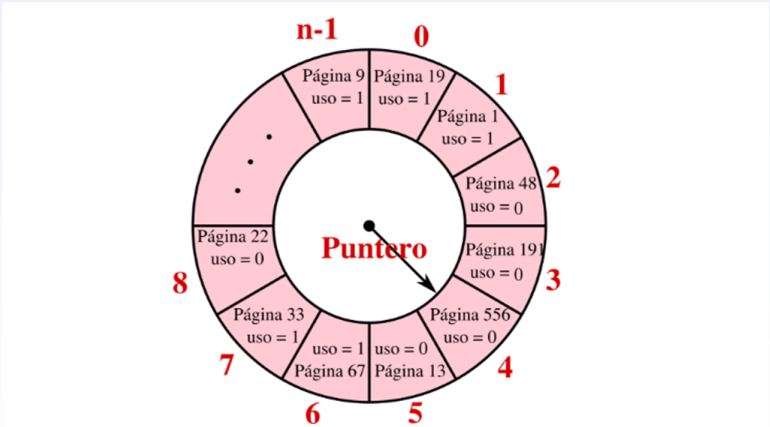
Del reloj

Ejemplo: página 1 recién cargada en marco 1 (referenciada). Avanza el puntero: la siguiente vez se comenzará buscando a partir del marco 2.



Del reloj

Busca desde marco 2. Da segunda oportunidad a páginas en marcos 2 y 3 (pone sus bits de uso a 0). Encuentra página con U=0 en marco 4 (en él cargará la página, pondrá bit de uso a 1 y avanzará al marco 5 para la siguiente búsqueda).



NRU (Not Recently Used)

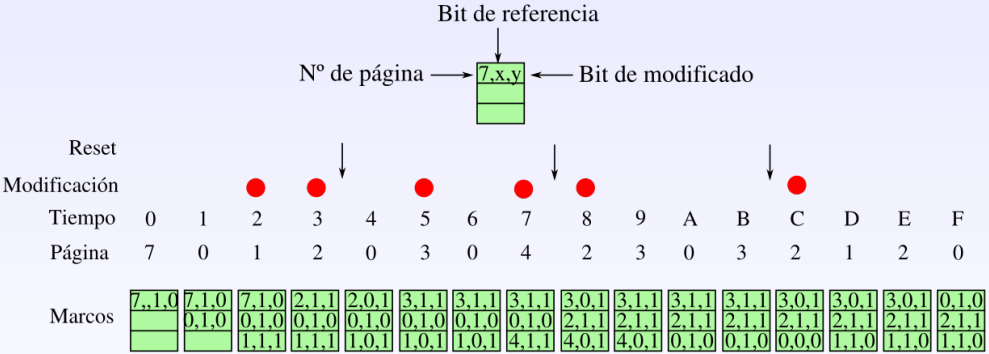
Utiliza los bits de Uso (U) y Modificado (M) de las entradas de la tabla de páginas. Se definen cuatro clases de páginas.

- **Clase 0**: Páginas no referenciadas recientemente ni modificadas (U=0,M=0).
- **Clase 1**: Páginas no referenciadas recientemente, modificadas (U=0,M=1).
- **Clase 2**: Páginas referenciadas recientemente, no modificadas (U=1,M=0).
- **Clase 3**: Páginas referenciadas recientemente, modificadas (U=1,M=1).

Algoritmo de selección de página a reemplazar:

- Buscar entre las de clase 0 (U=0, M=0)
- Si no se encuentra ninguna (el puntero ha dado la vuelta entera), buscar de clase 1 (U=0, M=1) y durante la búsqueda cambiar los U=1 que vaya encontrando por U=0.
- Si no se encuentra ninguna, buscar de nuevo de clase 0.
- Si no encuentra ninguna, buscar de nuevo de clase 1 (alguna encontrará, pues todos los bits de uso se han puesto a 0 en la 2a vuelta).

NRU (Not Recently Used)



NFU (Not Frequently Used)

- Detalle implementación: sigue una aproximación del algoritmo LRU, utilizando exclusivamente Sw:
  - Se activa un contador software por cada página
  - En cada interrupción de reloj, el SO suma el bit R (0 ó 1) al contador.
  - Cuándo existe un fallo de página se selecciona la página con el contador más bajo y se envía a disco.

R	contador	R	contador	R	contador	R	contador
1	00000001	1	00000011	1	00000111	1	00001111
0	00000000	1	00000001	1	00000011	1	00000111
1	00000001	0	00000010	0	00000100	0	00001000
0	00000000	0	00000000	1	00000001	1	00000011
1	00000001	1	00000011	0	00000110	0	00001100
1	00000001	0	00000010	1	00000101	1	00001011
	T=1		T=2		T=3		T=4

- **Problema**: Tiene memoria

NFU (Not Frequently Used)
NFU con maduración

- **Solución al problema de memoria**: Maduración.
  - Se activa un contador software por cada página.
  - En cada interrupción de reloj se desplaza el contador un bit a la derecha y luego se suma el bit R al bit más significativo del contador.

R	contador	R	contador	R	contador	R	contador
1	10000000	1	11000000	1	11100000	1	11110000
0	00000000	1	10000000	1	11000000	1	11100000
1	10000000	0	01000000	0	00100000	0	00010000
0	00000000	0	00000000	1	10000000	1	11000000
1	10000000	1	11000000	0	01100000	0	00110000
1	10000000	0	01000000	1	10100000	1	11010000
	T=1		T=2		T=3		T=4

NFU (Not Frequently Used)
NFU con maduración

Ventaja de reloj modificado (NFU) sobre reloj

Entre las páginas menos usadas, las no modificadas se reemplazan antes que las modificadas, lo que implica un ahorro de tiempo de escritura en disco (aunque una página se reemplace, no es necesario volver a escribirla en disco, pues no ha sido modificada).

En ambos, reloj y reloj modificado o NRU:

Al tener en cuenta el uso de las páginas, aunque haya que escribir en disco una página que ha sido reemplazada, se supone que dicha página tardará en ser usada de nuevo (principio de cercanía)

Almacenamiento Intermedio de páginas

- Estrategia para mejorar rendimiento de paginación.
- Creación de dos listas:
  - Lista de páginas libres = páginas sin modificar reemplazadas" (marcos libres)
  - Lista de páginas modificadas reemplazadas"
- La página a reemplazar en realidad no se quita de la memoria principal:
  - Se pone su bit de presencia=0 en su entrada en la tabla de páginas
  - Se pone en la lista de páginas libres o modificadas

Ventajas del almacenamiento intermedio de páginas

- Reduce el coste de cargar páginas que han sido reemplazadas hace poco tiempo
- Lista de páginas libres = lista de marcos disponibles para cargar páginas (se mantiene un mínimo)
- Las páginas modificadas son reescritas en disco por bloques en vez de una a una, reduciendo el número de E/S y el tiempo de acceso al disco.

Bloqueo de marcos

Algunos marcos de la memoria principal pueden bloquearse imponiendo una restricción a la política de reemplazo

- La página situada en un marco bloqueado no puede ser reemplazada
- Se utiliza para el núcleo del S.O. y otras áreas críticas
- Se asocia un bit de bloqueo a cada marco (en tabla de marcos o tabla de páginas)

Parte V

Criterios de diseño de paginación

Asignación de marcos en sistemas monoprogramados

- Se asignan páginas al SO y las restantes páginas libres se van asignando tras los correspondientes fallos de página a las páginas del proceso
- Una vez llenas todas las páginas, el manejador utiliza uno de los algoritmos de asignación por demanda para intercambiar páginas de la memoria y habilitar huecos para las nuevas páginas.
- Variación: el SO reserva parte de su espacio (libre) para apoyar la paginación. El espacio reservado sirve para alamacenar temporalmente la página entrante mientras se selecciona la página que se va a intercambiar.

Número de marcos

- **Límite superior**: no se puede asignar más del total de marcos libres.
- **Límite inferior**: Número máximo de referencias necesarias para completar una instrucción: **Límite inferior** $= i_{max} + o_{max}(1 + n_{max})$
  donde:
  - i_{max}: Número máximo de palabras que pueden componer una instrucción.
  - o_{max}: Número máximo de operandos que puede necesitar una instrucción.
  - n_{max}: Numero máximo de referencias indirectas a memoria necesarias para extraer los datos empleados por la instrucción que más referencias utilice.
- Si hay indirección múltiple, se debe limitar el número de referencias indirectas, de modo que si se supera se produzca una trampa al SO.
- **Ejemplo**: Ej. PDP-11 tiene instrucciones de dos palabras con 2 operandos que pueden estar direccionados indirectamente (1 nivel):
  Límite inferior $= 2 + 2 \times (1 + 1) = 6$marcos

Número de marcos

Asignación de marcos en sistemas multiprogramados

- **Asignación equitativa**: Cada proceso de los n existentes reciben el mismo número de marcos (m/n) , de los m marcos del sistema.
- **Asignación proporcional**: El número de marcos asignados a un proceso es proporcional a su tamaño. Si s_i es el tamaño de memoria virtual solicitado por le proceso p_i , entonces el número de marcos asignados al proceso p_i , a_i , de entre los m marcos del sistema será:

$$max(a_i = m\frac{s_i}{\sum s_i}, \textbf{número mínimo de marcos})$$

  - Si varía el nivel de multiprogramación se recalcula (a la alta o a la baja) el número de marcos asignados a cada proceso.
- **Asignación con prioridades**: El número de marcos asignados a un proceso no depende del tamaño del mismo sino de su prioridad.

Tamaño de página

- Razones para escoger un tamaño pequeño:
  - Reduce la fragmentación interna.
  - Favorece la localidad (lo que se carga en memoria se ajusta a lo que se necesita).
- Razones para escoger un tamaño grande:
  - Reduce el tamaño de la tabla de páginas (sólo el 1 % del tiempo de Lectura/Escrituras de/a disco, se debe a la transferencia, el 99 % son los tiempos de latencia y posicionamiento).
  - Reduce el número de fallos de página.
- No hay acuerdo:
  - Intel 80386: 4Kb
  - Motorola 68030: variable entre 256 bytes y 32Kb
  - IBM/370: 2 ó 4Kb

Área de almacenamiento en disco

Lo más simple es asignar un área dedicada (inicialmente vacía), separada del sistema de directorios. Opciones:

1. Cada vez que se crea un proceso se reserva una zona del área de intercambio igual al tamaño de imagen del proceso. A cada proceso se le asigna la dirección en disco de su área de intercambio, que se almacena en la Tabla de Proceso.La lectura se realiza sumando el número de página virtual a la dirección de comienzo del área asignada al proceso.
2. Si los datos y/o la pila pueden crecer, es mejor reservar zonas zonas independientes. A cada zona se le asignan varios bloques.
3. No se asigna nada inicialmente. A cada página se le asigna su espacio en disco cuando se va a intercambiar, y el espacio se libera cuando la página vuelve a memoria. Problema: se debe llevar contabilidad en memoria (página a página) de la localización de las páginas en disco.

Administración de fallos de página

1. El HW hace un señalamiento al núcleo, que guarda el contador de programa (y a veces el estado de instrucción interrumpida)
2. Ejecuta una rutina en ensamblador que resguarda los datos volátiles (registros) y llama al SO como un procedimiento
3. El SO detecta el fallo de página e intenta determinar la página virtual requerida (normalmente contenida en un registro, o a partir del contador de programa almacenado)
4. El SO verifica que la página es válida y los permisos. En caso de que no sea válida o no se disponga de los permisos adecuados el SP envía una señal al proceso o lo elimina. En caso contrario el SO intenta encontrar un marco libre y si no lo hay ejecuta el algoritmo de reemplazo de páginas.
5. Si la página seleccionada ha sido modificada se escribe en disco. Durante ese tiempo el planificador da paso a otro proceso

Administración de fallos de página

6. El SO busca en disco la página solicitada y planifica una operación de disco para recuperarla. Durante el tiempo de carga, el proceso solicitante sigue suspendido, y el planificador puede dar paso a otro proceso de usuario. El marco se pone a "ocupado".
7. Al terminarse la carga de la página el SO actualiza las tablas de páginas del proceso(s) involucrados.
8. La instrucción que cometió el fallo se reinicia (o vuelve al estado en que quedó al ser interrumpida). El Contador de Programa se modifica para que apunte a esa instrucción.
9. El proceso que provocó el fallo se planifica de nuevo y el SO regresa a la rutina que lo llamó.
10. La rutina restaura los registros y demás información volátil y regresa al espacio de usuario para continuar la ejecución.

Administración de fallos de página
Hiperpaginación (Trashing)

- Definimos **conjunto de trabajo** como el número de páginas activas que un proceso tiene en un momento dado. Es el número **suficiente**, mayor al mínimo. Si el número de marcos disponibles es inferior al tamaño del conjunto de trabajo, se producirán frecuentes fallos de página (**hiperpaginación**).
- Un proceso **hiperpaginado** pasa más tiempo intercambiando páginas que ejecutándose, y puedo 'robar' páginas de otros procesos, provocando su hiperpaginación.
- **Consecuencia**: reducción drástica del uso de CPU. El planificador de procesos responde **incrementando el nivel de multiprogramación**. Este procesos se realimenta positivamente hasta que el sistema se desploma.

Administración de fallos de página
Hiperpaginación (Trashing)

- La hiperpaginación se limita si se limita el número de marcos que el proceso puede utilizar (Asignación local), y si se asigna a cada proceso un número de marcos suficiente.
- **Problema**: Cómo calcular el número suficiente de marcos que un proceso necesita.
- Para calcular el tamaño del conjunto de trabajo, se utiliza el **modelo de conjunto de trabajo**, basado en el **supuesto de localidad**, que dice que en la ejecución de un proceso, las páginas que se utilizan en cada momento se pueden agrupar en grupos que varían a lo largo de la ejecución del proceso. Ej: una llamada a una función provoca un cambio de localidad.

Parte VI

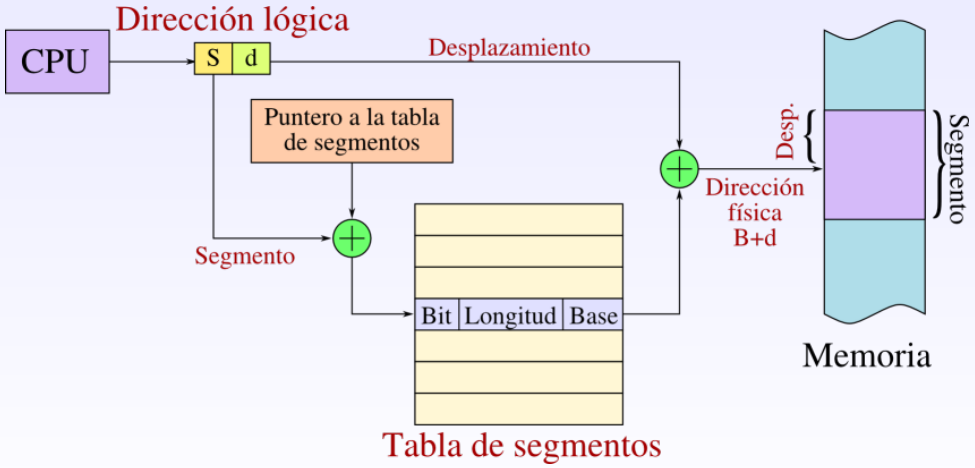
Segmentación

Segmentación

- Técnica para mantener espacios independientes de direcciones virtuales, llamados segmentos. Las distintas partes del proceso tienen espacios virtuales independientes entre sí.
- La segmentación permite:
  1. Asignar permisos distintos a las partes del proceso.
  2. Compartir ciertos datos o procedimientos entre procesos.
- Los segmentos se direccionan desde 0 hasta una dirección máxima, que puede variar.
- La segmentación pura puede producir problemas de fragmentación externa.

Segmentación

Proceso de segmentación

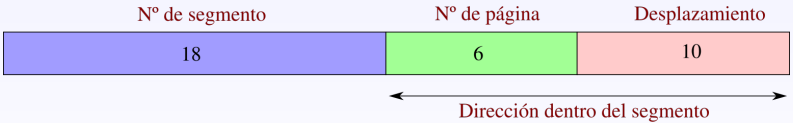


Segmentación paginada

- Segmentación + Paginación
  - Espacio de direcciones dividido en segmentos
  - Cada segmento dividido en páginas de tamaño fijo = tamaño de marco en memoria
  - Memoria principal dividida en marcos
- Si el segmento es menor que una página, ocupa un marco completo.
- Direcciones
  - Desde el punto de vista del programador: número de segmento + desplazamiento
  - Desde el punto de vista del sistema: número de segmento + número de página dentro de ese segmento + desplazamiento

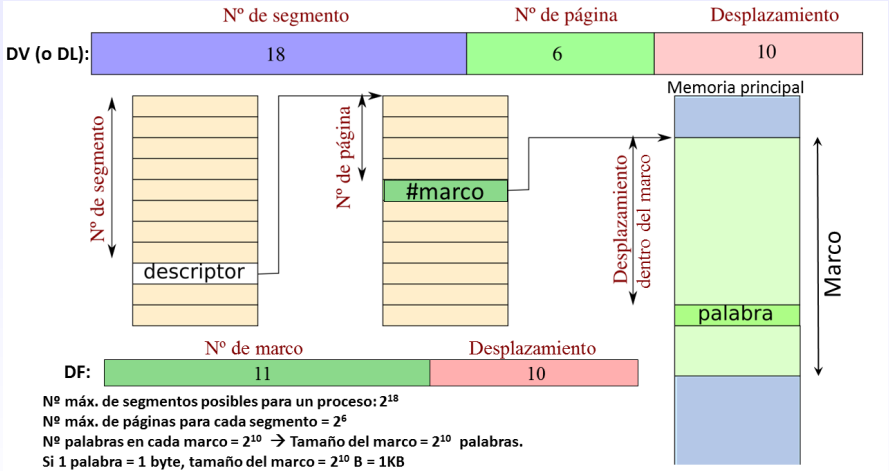
MULTICS (segmentación + paginación)

- Cada programa dispone de una memoria virtual de hasta 2^{18} segmentos, cada uno de hasta 64K (2^{18}) palabras. Cada segmento es un espacio virtual independiente que puede paginarse, con 64 (2^6) páginas de 1K (2^{10})
- Las direcciones físicas son de 24 bits, con las páginas alineadas con fronteras de 64 bytes
- La dirección virtual en MULTICS es:



MULTICS (segmentación + paginación)

Conversión de direcciones:



Nº máx. de segmentos posibles para un proceso: 2^{18}

Nº máx. de páginas para cada segmento = 2^{6}

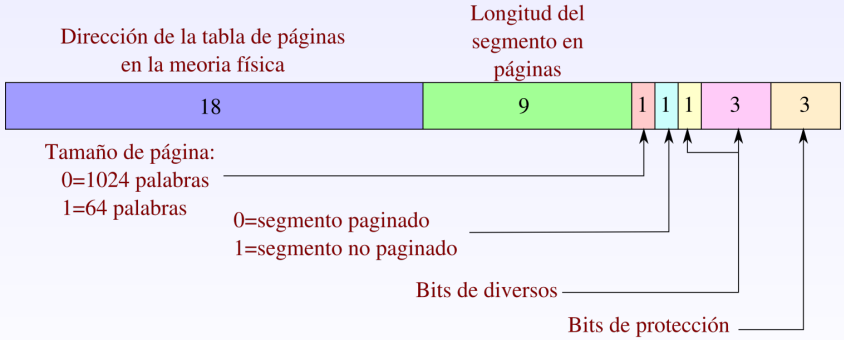
Nº palabras en cada marco = 2^{10} → Tamaño del marco = 2^{10} palabras.

Si 1 palabra = 1 byte, tamaño del marco = 2^{10} B = 1KB

Si memoria principal 2MB, nº marcos = 2MB/1KB = 2^{11} → 11 bits para #marco en la tabla de páginas y en la dirección física.

MULTICS (segmentación + paginación)

- Cada programa mantiene una tabla de segmentos
- Cada entrada de la tabla se segmentos tiene los siguientes campos:



Parte VII

Diseño y políticas del gestor de memoria

Diseño del gestor de memoria

Dependencias

1. Uso o no de memoria virtual (`Si hay hardware disponible`)
2. Uso de paginación, segmentación o ambas (`Si hay hardware disponible`)
3. Algoritmos empleados para los problemas de gestión de memoria

Información

- Actualmente, casi todos los SS.OO. Ofrecen Memoria Virtual (DOS y primeros UNIX no)
- Al usar segmentación paginada, la mayoría de los problemas surgen en la paginación (memoria principal=marcos).