

Tema 3. Gestión de memoria

1. Gestión de memoria principal	2.
1.1. jerarquía de memoria	2
1.2. espacio de direcciones lógicas y espacio de direcciones físicas	2
1.3. Gestión de memoria	3
2. Memoria Virtual: Organización	4.
2.1. Zonificación de memoria	4
2.2. Página	4
2.3. Segmentación	6
2.4. Segmentación paginada	7.
3. Memoria Virtual: Gestión	7
3.1. Influencia del tamaño de página	7
3.2. Algoritmos de sustitución	8
3.3. Comportamiento de los programas	9
3.4. Propiedad de localidad	9
3.5. Teoría del Conjunto de Trabajo	10
4. Gestión de memoria en Linux	11
4.1. Gestión de memoria a bajo nivel	11

1. Gestión de memoria principal

1.1. jerarquía de memoria

Hay dos principios clave:

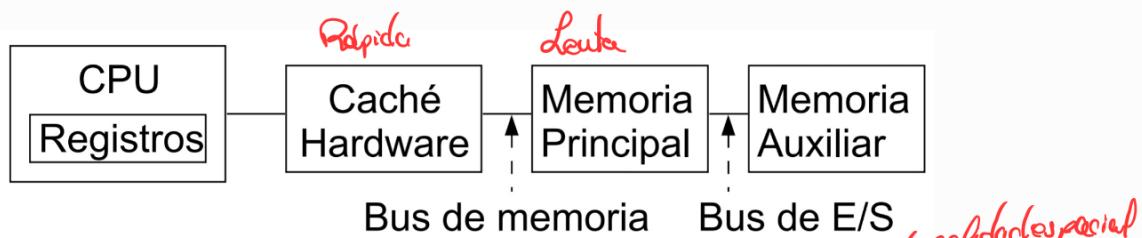
- Mayor cantidad → mayor velocidad
- Mayor cantidad → menor coste por byte

{ buscar un equilibrio

Debido a esto, los datos de acceso frecuente se ponen en memoria rápida, por ende costo y memoria pequeña, por ende costo y memoria lenta grande y barata.

→ caché

→ RAM



La memoria **caché** es aquella que contiene **los datos más frecuentes**, tiene mucha **velocidad de acceso**, de manera que si un dato es reciente, se obtiene de la caché; en cada acceso hay dos casos:

- **Acierto** (se encuentra en caché)
- **Fallo** (no está en caché y debe accederse a memoria principal)

Por cada acceso hay un tiempo transcurrido:

$$TAC = \% \text{ acierto} \cdot \text{coste de acierto} + \% \text{ fallo} \cdot \text{coste de fallo.}$$

Esta organización es muy útil pues explota el **principio de localidad**. En memoria caché, no solo se encuentran datos pedidos, sino todo un entorno cercano a los mismos, reduciendo así la probabilidad de fallo.

1.2. Espacio de direcciones lógicas y espacio de direcciones físicas

El **espacio de direcciones lógicas** es el conjunto de direcciones que se generan dentro de un programa, la línea 0 de un programa en dirección lógica puede ser la dirección de memoria 0x86, es decir, 0x86 es su dirección física.

Direcciones lógicas y físicas no tienen por qué coincidir.

Por tanto, el espacio de direcciones físico es el conjunto de direcciones de memoria principal correspondientes a un espacio de direcciones lógico.

Relacionado con esto encontramos el mapa de memoria de un proceso; no es más que el "entorno" creado por la CPU para la ejecución de un proceso, donde se copia código, datos con valor inicial, datos sin valor inicial e inicialmente un fragmento de pila es reservado. Todo esto es generado a partir del fichero ejecutable, dividido en cabecera y secciones.

La imagen de proceso se calcula como la suma del mapa y del PCB

1.3 Gestión de memoria

Problemas:

- División de la memoria
- Estrategias para el rendimiento óptimo
- Protección de la memoria y del SO.

Estrategias de rendimiento óptimo

La asignación de memoria a un proceso puede hacerse de dos formas:

- **Contigua:** Cada programa recibe un bloque de almacenamiento único de posiciones secuenciales o contiguas en memoria; dichos bloques pueden ser fruto de particiones de tamaño fijo o de tamaño variable.
- **No contigua:** Permite fragmentar un proceso en varias partes y asignar en memoria su bloque que lo tiene por que estar contiguo al anterior. Son los más vendidos: paginación, secuenciación y segmentación paginada.

Para ellas, hay que hacer uso del intercambio de memoria swap, también considerada junto a la principal memoria virtual; esto supone el cambio entre intercambiar procesos entre memoria y el almacenamiento auxiliar que debe ser unidireccional con espacio para albergar las imágenes de memoria de los procesos de usuario.

Este intercambio requiere un tiempo y es realizado por el intercambiador.

- Selecciona procesos para retirarlos de MP.
- Selecciona procesos para incorporarlos a MP.
- Gestionar y asignar el espacio de intercambio.

Fichero Ejecutable	
0	Cabecera
4	
...	
96	LOAD R1, #1000
100	LOAD R2, #2000
104	LOAD R3, /1500
108	LOAD R4, [R1]
112	STORE R4, [R2]
116	INC R1
120	INC R2
124	DEC R3
128	JNZ /12
132
136	

Carretero, p.165

2. Memoria virtual: Organización

La memoria virtual no es más que la memoria principal junto a un dispositivo de almacenamiento ~~externo~~ ^{externo} que va a "asumir" nuestra memoria principal. Es a ese conjunto al que denominamos memoria virtual.

En memoria principal se guardan las partes del proceso necesarias en un momento dado, sin embargo, en memoria secundaria está guardado el espacio de direcciones completas del proceso.

La gestión de esta memoria implica conocer qué hay en memoria principal y una política de movimiento entre memoria principal y swap.

Como ventajas, resuelve el problema del crecimiento dinámico de los procesos y permite aumentar el grado de utilización.

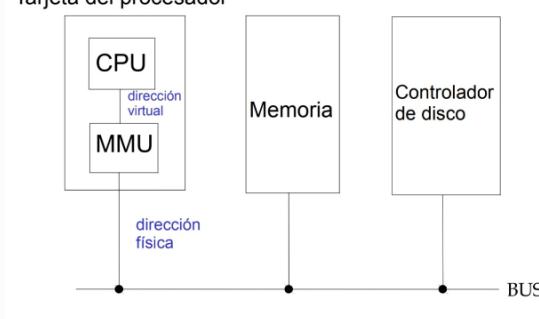
2.1. Unidad de gestión de memoria

El MMU es el dispositivo hardware que se encarga de traducir direcciones virtuales a direcciones físicas siendo controlado por el sistema operativo; por tanto es un componente imprescindible para cargar un programa, localizado en swap, en memoria principal.

Debe destacar que un programa de usuario sólo habla con direcciones lógicas, por lo que el MMU es el encargado de traducir esas direcciones.

Además, en cada traducción, la MMU es la encargada de comprobar si su traducción ha sido correcta oprobando si existe en memoria principal; en caso contrario, lanzará una excepción.

Tarjeta del procesador



Cada MMU tiene a su disposición el búfer de traducción adentrado o TLB, donde se encuentra un grupo de direcciones lógicas junto con su traducción, reduciendo así el tiempo de traducción y facilitando el trabajo al MMU.

2.2. Paginación

Consiste en hacer divisiones en cuadros de memoria, tanto física como virtual y lógica.

En este caso, las divisiones de tamaño fijo de memoria virtual se les llama marcos o página y suele ser potencia de 2. Si las divisiones del espacio lógico de tamaño fijo se les llaman páginas.

En este sistema, los marcos de página contienen la página. No es necesario, pero sí útil, que cada marco solo contenga una página.

Las direcciones lógicas las genera la CPU y se dividen en nº de página y desplazamiento, por otro lado, las direcciones físicas se dividen en número de marco de página y desplazamiento. Será el mismo que en las direcciones lógicas.

Cuando la CPU genera una dirección lógica, será necesario traducirla a la dirección física correspondiente. Para ello, el sistema necesita una estructura llamada tabla de páginas que almacena la información necesaria para dicho cambio. Es única por proceso.

Además, para facilitar el swapping, hay una tabla de ubicación en disco, que guarda la ubicación en disco de cada página de un proceso en swap; debido a esto es única por proceso.

Por último, hay una tabla de marcos de página que es usada por el SO y contiene información sobre cada marco de página.

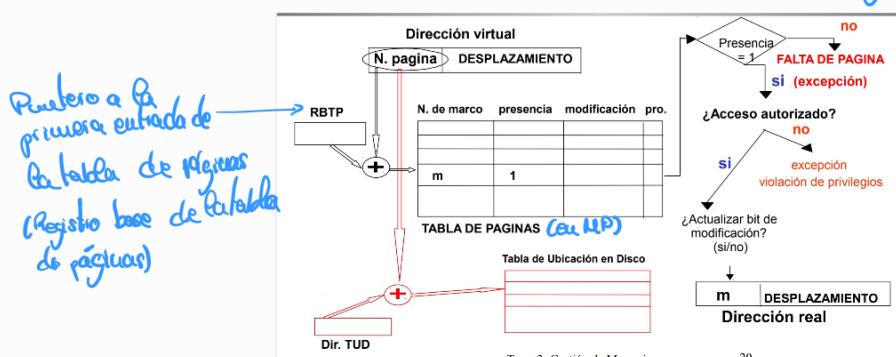
Tabla de páginas

Tiene una entrada por cada página de un proceso donde en cada entrada se almacenan:

- Nº de marco donde está la página en MP
- Bit de presencia
- Bit de modificación
- Privilegios de la página

Nº de página	nº marco	presencia	modificación	protección
	46	1	0	01

Para traducir de direcciones virtuales a reales se sigue el siguiente esquema:



Cuando se produce una falta de página, se siguen los siguientes pasos:

1. Identificar proceso
2. Encuentra la ubicación en swap de la página
3. Encuentra marco libre, si no hay, se descarga
4. Cargar página desde disco
5. Actualizar tablas
6. Desplazar proceso
7. Reiniciar la instrucción que ocasionó la falta de página.

Si seguimos el enfoque propuesto, cada acceso requiere dos accesos a memoria, uno a la tabla de páginas y otro a memoria (resuelto con TLB). Sin embargo, se nos presenta el problema del buzamiento de la tabla de páginas.

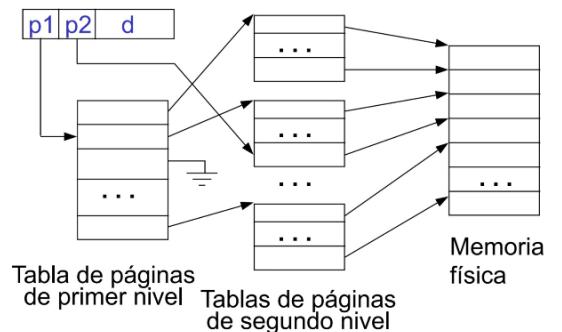
Para solventarlo, usaremos páginaación multilevel, es decir, página las tablas de páginas, de manera que sólo esté cargada en memoria la partición que se está realmente usando. Los demás se dejan en memoria secundaria. Sólo cargo lo que necesito.

Para esto:

- Dirección lógica : n bits número de página $\xrightarrow{\text{u bits en número de página}}$ $n-u$ bits desplazamiento de página

n bits desplazamiento de página.

La tabla de páginas se divide en fragmentos del buzamiento de una página.



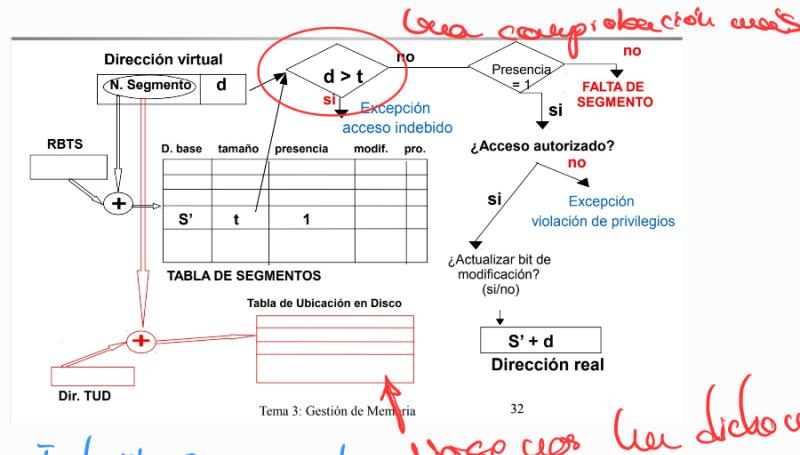
Hay ocasiones donde varios procesos usan un mismo código reentrante, dando lugar a que una misma página esté cargada多次 en MP; sin embargo, esto no es así, esta página es compartida por ambos procesos. Esta práctica puede ocasionar fallos de protección.

2.3. Segmentación

Consiste en el caso que páginaación pero tomaendo páginas de tamaños variables, por tanto, cada tabla desegmentada habrá un campo adicional con el tamaño desegmento.

En cada fragmento de dirección lógica, deberá comprobarse que el desplazamiento es menor al tamaño de segmento.

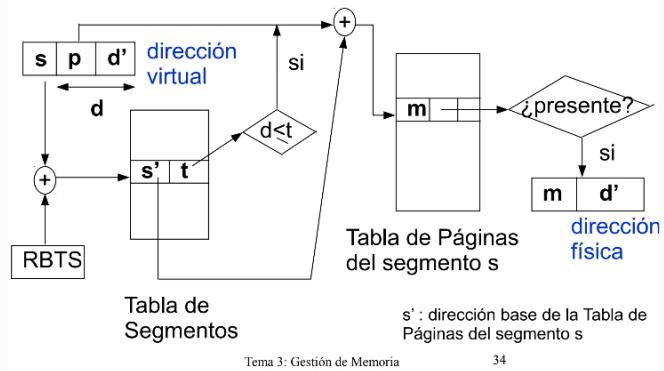
Además del registro base de la tabla de segmentos, dispuestos de registro base/bound de la tabla de segmentos que guarda la cantidad de segmentos de un proceso; por tanto el o de segmento de una dirección lógica deberá ser menor a este registro.



Traducción en segmentación *No se usa el disco duro*

2.3. Segmentación Página

Consiste en hacer una gestión de memoria basada en segmentos, pero dividir cada segmento en páginas; ganando así las beneficios de la segmentación y eliminando los problemas de una gestión de memoria compleja.



3. Memoria Virtual: Gestión

Normalmente, dicha gestión es realizada con paginas y se siguen una serie de criterios de clasificación:

- Políticas de asignación: Fija o variable
- Políticas de búsqueda: Por demanda o anticipada
- Políticas de sustitución: Local o global.

Independientemente de la política de sustitución usada, siempre debe cumplirse:

- Limpia antes que sucias
- Uso de páginas compartidas elevado
- Uso de páginas especiales

3.1. Influencia del tamaño de página

Páginas pequeñas:

- Tablas de páginas grandes.
- Mayor número de tránsferencias LRP → Disco
- Menor frecuencia de referencia.

Páginas grandes:

- Mucha información está ocupando memoria principal cuando se va a usar.
- Mayor fragmentación interna.

Como solución, se busca un **término medio**.

3.2. Algoritmos de sustitución

Categorías:

Asignación	Sustitución
Fija	Local
Variable	Local
Variable	Global

Fija - global \cancel{A}

Algoritmo óptimo

Sustituye la página que **se va a referenciar en un futuro** a lo que se va a referenciar más tarde.

Este algoritmo supone la utilización de algún revisor para hacer esa predicción; que a priori, no tiene por qué ser correcta. \rightarrow No se usa.

Algoritmo FIFO

Sustituye la **página más antigua**, no tiene en cuenta la frecuencia de referenciado a una página.

Algoritmo LRU (Reciente Menos Usada)

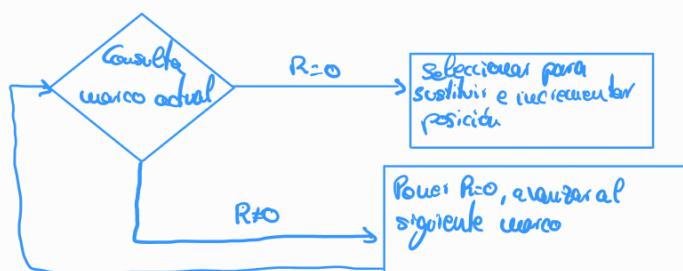
Sustituye la **página que fue objeto de la referencia más antigua**.

Ninguno de los anteriores se usa en la actualidad, el más usado es el algoritmo del reloj.

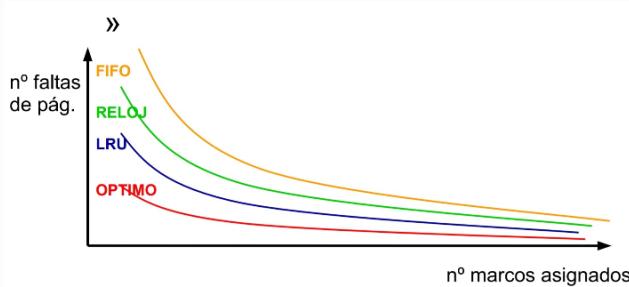
Algoritmo de reloj:

Cada página tiene asociado un **bit de referencia** R que será inicializado a 1 por el hardware; los marcos de página son representados por una **lista circular** y con punto a la **siguiente página** más tiempo:

Funcionamiento:

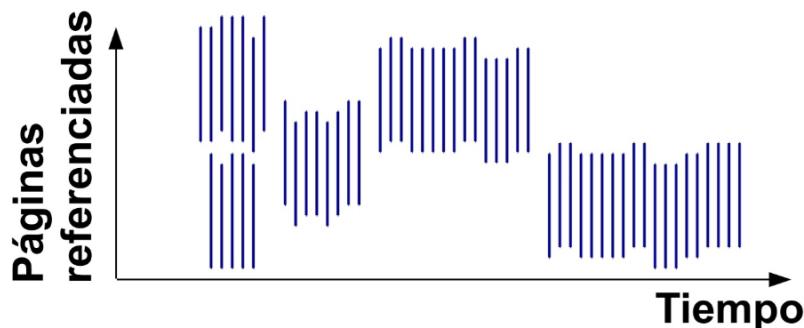


Para reducir el número de faltas producidas, no influye realmente el algoritmo usado, sino la cantidad de marcos disponibles.



3.3. Comportamiento de los programas

Este viene definido por la secuencia de referencias a páginas que realiza un proceso, dicho establecimiento debe realizarlo para maximizar el rendimiento del sistema de memoria virtual.



3.4. Propiedad de Localidad

Los principios de localidad deben ser uno de los criterios más importantes al trabajar con memorias cache.

→ **Temporal**: Una posición de memoria referenciada recientemente tiene una alta probabilidad de serlo de nuevo.

→ **Espacial**: Las posiciones cercanas a una referenciada pueden ser referenciadas posteriormente.

Respecto a esto, se define el **conjunto de trabajo** de un proceso como el grupo de páginas que son altamente referenciadas en un intervalo de tiempo.

$$WS(t_1, t_2) = \{ \text{páginas referenciadas en el intervalo } [t_1, t_2] \}$$

Con propiedades:

(buzón)

Es un buzon servido por el tiempo futuro.

- Son transitorios
- No se puede predecir el futuro de un conjunto de trabajo
- Diferen unos de otros sustancialmente.

3.5. Teoría del Conjunto de Trabajo

Este teorema cumple dos postulados:

- i) Un proceso solo puede ejecutarse si su conjunto de trabajo es en memoria principal ↓ Recinto
- ii) Una página permanecerá en MP mientras pertenezca a un conjunto de trabajo

Un problema a resolver es la hiperpaginación, es decir, cuando un proceso tiene suficientes páginas, la tasa de fallos de página es alta lo que provoca un bajo uso de CPU, un aumento del grado de multiprogramación y más fallos de página.

Por tanto, esto provoca que el so esté "distraído" produciendo fallos de página.

Para solucionarlo, hay dos formas:

- diseñar, a cada proceso, un espacio en relación a su comportamiento → diseñar su variable
- actuar directamente sobre el grado de multiprogramación → Regulación de carga

Algoritmo basado en el modelo del WS

En cada referencia se calcula el conjunto de trabajo WS(t,x) y sobre esas páginas se actualizan en MP

Cada falta de página se simboliza con un asterisco (*)

Algoritmo FFP

Se usa el intervalo de tiempo entre las faltas de página consecutivas:

- Si el intervalo es grande, > 5 , todas las páginas no referenciadas en dicho intervalo son retiradas de MP. → se quitan las no referencias.
- Si el intervalo es pequeño, ≤ 5 , simplemente se incluye el conjunto de páginas residente. → se añade

Este algoritmo garantiza que el conjunto de páginas residentes aumenta cuando las faltas de página son frecuentes y disminuye cuando son bajas.



4. Gestión de memoria en Linux

4.1 Gestión de memoria a bajo nivel

La página física es la unidad básica de gestión de memoria. Unidades de página de 1 página

```
struct page {  
    unsigned long flags; // PG_dirty, PG_locked  
    atomic_t _count;  
    struct address_space *mapping;  
    void *virtual;
```

Dichas páginas pueden ser usadas por la cache de páginas, datos privados, una proyección de la tabla de páginas de un proceso, el espacio de direcciones de un proceso, datos del kernel a los que directamente o código kernel.

También proporciona funciones para asignar páginas y liberarlas, incluso en bloques.

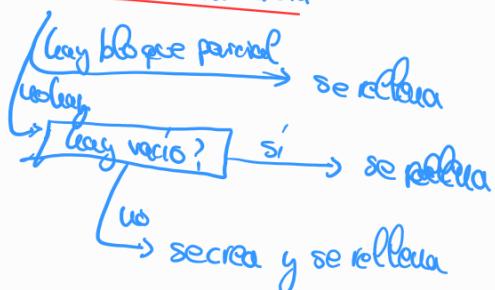
Selinux, dispone de zonas de memoria, para poder especificar el tipo dispone de tipo GFP_T. Esta liberación/asignación es realizada por Linux en bloques ademas, el nivel de bloque actua como nivel de cache de estructuras generadas:

- Una cache por tipo de estructura distinta
- Multiples bloques de una o mas páginas contiguas para ello
- Cada bloque contiene información tipo de la cache

Funcionamiento:

: Tres estados: lleno, parcial o vacío

→ Se lanza la nueva estructura



Cada proceso requiere del espacio de direcciones cuando está ejecutándose, en Linux, se usa memoria virtual; de manera que cada proceso solo tiene acceso a su área asignada. Para gestionarlo, se usa el descriptore memoria.

Cada archivo tiene almacenado como atributos:

- start-end
- Permisión
- Offset
- major:minor: Son los números mayor y menor del dispositivo.
- cuando: no definido
- file: nombre del archivo

Cuando gestiona memoria usa paginación y es a tres niveles.

Hablando de la cache de páginas, cabe destacar:

- Confina páginas físicas de RAM.
- Tamaño fijo.
- Dispone de un área de respaldo.
- Se lee o escribe dentro o en disco.

Para recuperar y eliminar datos de cache, sólo realiza un recambio de páginas lru (menos usados por PG-dirty). Si no los hay, se mandan algunas a disco para que las baje al disco y se selecciona la página a eliminar, para ello usa el algoritmo CRU. Por último, es posible que cada página contenga varios bloques.