

tema3.pdf



postdata9



Sistemas Operativos



2º Grado en Ingeniería Informática



**Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada**

WUOLAH



El más PRO del lugar puedes ser Tú.



¿Quieres eliminar toda la publi de tus apuntes?



¡Fuera Publi!
Concéntrate al máximo



Apuntes a full.
Sin publi y sin gastar coins

Para los amantes de la inmediatez, para los que no desperdician ni un solo segundo de su tiempo o para los que dejan todo para el último día.

Quiero ser PRO

4,95 / mes

Tema 3:

Gestión de memoria

ÍNDICE:

- 1. Introducción**
- 2. Organización de la Memoria Virtual**
 - 2.1 Concepto de Memoria Virtual**
 - 2.2 Paginación**
 - 2.2.1 Contenido de la tabla de páginas**
 - 2.2.2 Esquema de traducción**
 - 2.2.3 Falta de página**
 - 2.2.4 Implementación de la Tabla de Páginas**
 - 2.2.5 Tamaño de la Tabla de Páginas**
 - 2.2.6 Paginación multinivel**
 - 2.2.7 Páginas compartidas**
 - 2.3 Segmentación**
 - 2.3.1 Tabla de Segmentos**
 - 2.3.2 Esquema de traducción**
 - 2.4 Segmentación paginada**
 - 2.4.1 Esquema de traducción**
- 3. Gestión de la Memoria Virtual**
 - 3.1 Introducción**
 - 3.1.1 Políticas de asignación**
 - 3.1.2 Políticas de sustitución (o de reemplazo)**
 - 3.1.3 Políticas de búsqueda (o de recuperación)**
 - 3.1.4 Influencia del tamaño de página**
 - 3.2 Algoritmos de sustitución**
 - 3.2.1 Algoritmo Óptimo**
 - 3.2.2 Algoritmo FIFO**
 - 3.2.3 Algoritmo LRU**
 - 3.2.4 Algoritmo del reloj**
 - 3.2.5 Comparación**
 - 3.3 Comportamiento de los programas**
 - 3.3.1 Propiedad de localidad**
 - 3.3.2 Conjunto de Trabajo**
 - 3.3.3 Teoría del Conjunto de Trabajo**
 - 3.4 Hiperpaginación (Thrashing)**
 - 3.5 Algoritmos de asignación variable y sustitución global**
 - 3.5.1 Algoritmo basado en el modelo del WS**
 - 3.5.2 Algoritmo Frecuencia de Falta de Página (FFP)**

1. Introducción

Los objetivos generales de la gestión de memoria son:

- **Organización:** cómo divide el SO la memoria.
- **Gestión:** a partir de una organización, se ve cómo existen estrategias que ayudan en el núcleo a obtener un rendimiento óptimo; son códigos que implementan:
 - **estrategias de asignación** de memoria a los procesos;
 - **estrategias de sustitución** cuando hay necesidad de que un proceso sea sustituido por otro;
 - **estrategias de búsqueda**, de soluciones concretas a problemas concretos.
- **Protección:** de la memoria.

El sistema mediante la **memoria virtual** intenta mostrar a los procesos que tiene **más memoria física** de la que tiene, lo que se consigue con la **extensión de la memoria principal** a la memoria de **almacenamiento secundario**, como discos, memorias rápidas. Esto conlleva que el sistema realice tareas de llevar y recuperar segmentos/páginas de procesos a disco.

El sistema intenta mantener un **espacio de memoria principal libre** para que en algún momento de demanda del sistema, se garantice una **respuesta en un tiempo razonable**.

Intercambio (swapping)

- **Intercambiar** procesos entre memoria y un almacenamiento auxiliar.
- El almacenamiento auxiliar debe ser un **disco rápido** con espacio para albergar las imágenes de memoria de los procesos de usuario.
- El factor principal en el tiempo de intercambio es el **tiempo de transferencia**.
- El intercambiador tiene las siguientes responsabilidades:
 - Seleccionar procesos para **retirarlos e incorporarlos** a memoria principal.
 - **Gestionar y asignar** el espacio de intercambio.

En Linux, hay hebras del núcleo implementadas que realizan algunas de estas tareas.

En el **espacio de intercambio (swapping)** hay dos soluciones:

1. En un sistema de archivos existente, hay un **archivo del sistema especial**, como el archivo swap, que se dedica únicamente a **albergar contenido de memoria**, páginas de los procesos que se están ejecutando y no están en memoria principal.
2. Existe un espacio de intercambio, que es una **partición dedicada al intercambio**. No es un archivo dentro del sistema de archivos, sino una partición que su sistema de archivos es muy básico, no hay metainformación, solo alberga las páginas que no caben en memoria principal.

En su momento, el swapping supuso una manera de permitir que en los sistemas del tiempo compartido existieran más procesos de los que caben en memoria. Se puede considerar el antecesor de la memoria virtual.

2. Organización de la Memoria Virtual

2.1 Concepto de la Memoria Virtual

La memoria virtual es una técnica de **gestión de la memoria** que se encarga de que el SO disponga de la mayor cantidad de memoria que esté disponible físicamente, tanto para el software de usuario como para sí mismo.

El tamaño del programa, los datos y la pila **puede exceder** la cantidad de memoria física disponible para él. Cuando hay que ejecutar un programa y se convierte en proceso por parte del sistema para darle contexto y demás, aunque no cupiera en memoria principal por su tamaño, el sistema sería capaz de ejecutarlo: esta es la **idea principal de memoria virtual**.

El sistema le da la sensación a los procesos de que tiene espacio en memoria principal.

Esto se realiza usando un **almacenamiento a dos niveles**:

- **Memoria Principal** → partes del proceso necesarias en un momento dado.
- **Memoria Secundaria** → espacio de direcciones completo del proceso. Con archivos o particiones de intercambio (swap).

Los **aspectos principales** de la memoria virtual son:

- las direcciones generadas por las instrucciones máquina forman parte del mapa de memoria virtual. Se dice que el procesador genera “**direcciones virtuales**”;
- para que el programa pueda ejecutarse, los **datos** y las **instrucciones** deben residir en **memoria principal**, por lo que es necesario una **migración de información** entre el disco y la memoria principal;
- los espacios virtuales y físicos se dividen en **páginas**:
 - **páginas virtuales**: son las páginas de espacio virtual;
 - **páginas de intercambio**: son las páginas residentes en disco;
 - **marcos de página**: son los espacios en los que se divide la memoria principal.

Un marco de página es capaz de albergar una página virtual.

La memoria virtual puede:

- resuelve el problema del crecimiento dinámico de los procesos, por ejemplo, la pila, con llamadas recursivas, con lo que la página aumenta;
- puede aumentar el grado de multiprogramación, al no ser necesario que todo el mapa de memoria de un proceso esté en memoria principal para poder ejecutarse;
- ejecutar programas más grandes que la memoria principal disponible.

Para ello, necesita:

- saber qué se encuentra en memoria principal;
- una política de movimiento entre memoria principal y memoria secundaria.

Esta técnica no es apropiada para sistemas en tiempo real, debido a que puede ralentizar la sobrecarga asociada a las transferencias entre la MP y MS.

¿BUSCAS LIBROS O EBOOKS SOBRE ENFERMERÍA?

AXON librería especializada en Ciencias de la Salud.

AXON
axon.es



Encuentra todos los libros y eBooks para tu especialización, además del material complementario que necesites.

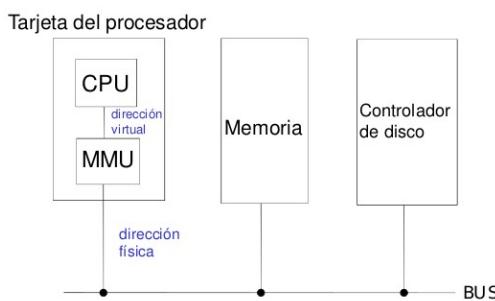
Unidad de Gestión de Memoria

La MMU (Memory Management Unit) es un dispositivo hardware que traduce rápidamente direcciones virtuales a direcciones físicas y está gestionado por el SO.

Cuando el sistema va a acceder a la dirección, se realiza esta traducción. Al ser por hardware es más rápido.

- En el esquema MMU más simple, el valor del registro base se añade a cada dirección generada por el proceso de usuario al mismo tiempo que es enviado a memoria
- El programa de usuario trata sólo con direcciones lógicas; éste nunca ve direcciones reales.
- Además de la traducción deberá:
 - disparar el manejador de falta de protección (una rutina del núcleo) con motivo de excepciones;
 - manipular la falta de validez, que es la ausencia de una página a la que quiere acceder el proceso pero no está actualmente en memoria principal.

Con estas dos rutinas básicas que tiene el núcleo a nivel de MMU, se asegura la protección.



2.2 Paginación

- El espacio de direcciones físicas de un proceso puede ser **no contiguo**.
- La **memoria principal física** se divide en bloques de tamaño fijo, denominados **marcos de página**. El tamaño debe ser **potencia de 2**, de 0.5 a 8 Kb. Los marcos de páginas contendrán páginas de los procesos.
- El **espacio lógico de un proceso** se divide en bloques del mismo tamaño, denominados **páginas**, es decir, que cada proceso se divide en páginas de igual tamaño, también potencia de 2.
- Las **direcciones lógicas**, que son las que **genera la CPU** se dividen en **número de página** (p) y **desplazamiento** dentro de la página (d).



- Las **direcciones físicas** se dividen en **número de marco** (m, dirección base del marco donde está almacenada la página) y **desplazamiento** (d).



- Cuando un proceso se trae a la memoria, todas sus páginas se cargan en los marcos disponibles, y se establece una tabla de páginas. Existe una **tabla de páginas por proceso**.
- Cuando la CPU genere una dirección lógica será necesario traducirla a la dirección física correspondiente, la tabla de páginas mantiene información necesaria para realizar dicha traducción.
- La tabla de páginas está asociada con un programa cargado en memoria y contiene una entrada por cada página del proceso en la que se indica el número de marco de la memoria (y más información que se verá después).

The diagram illustrates the state of memory management for four processes:

- Tabla de páginas del proceso A:** Contains pages 0, 1, 2, 3 at addresses 0, 1, 2, 3 respectively.
- Tabla de páginas del proceso B:** Contains pages 0, 1, 2 at addresses 0, 1, 2 respectively, with page 3 being invalid (-).
- Tabla de páginas del proceso C:** Contains pages 0, 1, 2, 3 at addresses 7, 8, 9, 10 respectively.
- Tabla de páginas del proceso D:** Contains pages 0, 1, 2, 3, 4 at addresses 4, 5, 6, 11, 12 respectively.
- Lista de marcos libre:** Shows free pages 13 and 14.

- El sistema operativo almacena una **tabla de marcos de página** para saber qué marcos están libres y cuáles están ocupados.
- Hay también dos tablas:
 - **Tabla de ubicación en disco:** ubicación de cada página en el almacenamiento secundario;
 - **Tabla de marcos de página:** usada por el SO y contiene información sobre cada marco de página.

2.2.1 Contenido de la tabla de páginas

Una tabla de páginas contiene todas las páginas que tiene un proceso; hay una tabla por proceso. Cada una de las entradas de esta tabla contiene:

- **Número de marco** (dirección base del marco): indica la dirección física del marco en el que está almacenada la página si está en memoria principal.
- **Bit de presencia o bit válido:** indica que la página se encuentra cargada en memoria principal, que no está en un dispositivo de intercambio.
- **Bit de modificación:** se suele llamar bit de modificado sucio y establece si se ha modificado la página.
- **Modo de acceso autorizado a la página (bits de protección):** indica qué puede realizar el proceso en dicha página, si leer, escribir, o leer y escribir.

Nº de página

nº marco	presencia	modificación	protección
46	1	0	01

El tamaño de la página debe ser potencia de 2:

- Un **tamaño pequeño** de página **reduce la fragmentación** y permite ajustarse mejor al conjunto de trabajo del proceso.
- Un **tamaño grande** implica **tablas más pequeñas** y cuando se usa para implementar memoria virtual supone un mejor rendimiento en los accesos a los discos.

La **tabla de segmentos** (tabla de regiones por proceso) tiene una entrada por cada uno de los **segmentos** que tiene el proceso: un segmento de texto, de código, de datos, de pila, al heap, etc.

Desde la tabla de regiones globales (la tabla a la que apunta cada una de las entradas de la tabla de regiones por proceso), no se va directamente a memoria, sino que llevaba a una **tabla de páginas**.

- El espacio de direcciones virtuales de un **Espacio de direcc. virtuales** proceso son las páginas que componen un segmento de un proceso en un momento dado. En el ejemplo, el segmento del proceso tiene 12 páginas.
- La tabla de páginas debe tener tantas entradas como páginas tenga el segmento.
- En el ejemplo se ve que:
 - la página 1 se encuentra en el marco 8 de memoria principal;
 - la página 2 no se encuentra cargada en memoria principal (bit de presencia 0) por lo que no tiene ningún marco asignado;
 - la página 11 no tiene ni marco ni bit de presencia porque dicha página no existe en el proceso.

Tabla de Páginas			Memoria Física
Nº de Marco	Bit de presencia	Num. Marco	
1	8	1	Pag3
2	-	0	Pag4
3	1	1	
4	2	1	
5	-	0	
6	-	0	
7	6	1	
8	3	0	
9	-	0	
10	5	1	
11	-	-	
12	10	1	

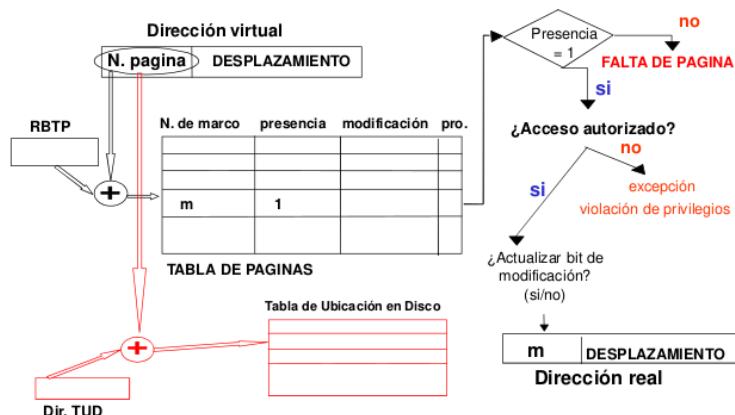
Num. Pagina →

2.2.2 Esquema de traducción

El hardware de gestión de memoria utiliza la tabla de páginas para traducir todas las direcciones que genera un programa.

La traducción consiste en detectar qué página se corresponde con una dirección lógica y acceder a la tabla de páginas para obtener el número de marco en el que se encuentra dicha página.

El sistema operativo mantiene una tabla por proceso y notifica al hardware en cada cambio de proceso qué tabla tiene que usar apoyándose en el registro base de la tabla de páginas (RBTP), que se almacena en el PCB del proceso.



El **esquema de traducción** sería el siguiente:

- El proceso referencia la **dirección lógica** generada por la CPU, que es **p:d**, es decir, el número de página y el desplazamiento.



- Se busca en la tabla de páginas la página **p**. La tabla de páginas asociada a cada proceso tiene la siguiente estructura:

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

Como se puede estar ejecutando más de un proceso a la vez, habrá varias tablas de páginas, por ello el acceso a la tabla de páginas de un proceso se realiza con ayuda del **registro base de la tabla de páginas** (RBTP) asociado a cada programa.

Este RBTP se almacena en el PCB de cada proceso.

- Si el bit de presencia es:
 - 0 → ocurre un **fallo de página**, con lo que hay que traer la página del disco a memoria principal y actualizar los datos en la tabla de páginas del proceso;
 - 1 → la página está en memoria principal, pero primero hay que comprobar que el proceso tiene permiso para acceder a dicha página.
 - No tiene permiso → ocurre una **excepción**, una violación de los privilegios de la página.
- Si tiene permiso, ya tenemos la **dirección física m:d**, donde **m** es el número de marco en memoria principal, y **d** es el desplazamiento, el mismo que el de la dirección lógica.



- Al tener ya la dirección física, se accede al marco **m** en memoria principal y se desplaza **d** en dicho marco para obtener la página para el proceso.

PANDURA



MOTIVACIÓN



CONCENTRACIÓN



MEMORIA

PANDURA



"Sin Pandora,
juegas en
desventaja."



"Memorizo un
50% más en el
mismo tiempo."



MELEENA DE LEÓN



BACOPA MONNIERI



CAMELLIA SINENSIS



GINGKO BILOBA



HUPERZIA SERRATA



COFFEA ARABICA

Realiza tu diseño sobre el tema de color negro.

2.2.3 Falta de página

Cuando falta una página:

1. Bloquear proceso.
2. Encontrar la ubicación en disco de la página solicitada (tabla de ubicación en disco).
3. Encontrar un marco libre. Si no hubiera, se puede optar por desplazar una página de MP.
4. Cargar la página desde disco al marco de MP.
5. Actualizar tablas (bit presencia=1, no marco, ...).
6. Desbloquear proceso.
7. Reiniciar la instrucción que originó la falta de página.

2.2.4 Implementación de la Tabla de Páginas

A pesar de sus ventajas y correcto funcionamiento, la **paginación** conlleva **dos problemas** derivados de mantener las tablas de páginas en memoria principal:

- **Eficiencia:** para consultar la entrada de la tabla, se deben realizar **dos accesos a memoria** por cada acceso real solicitado, es decir, un acceso a la tabla de páginas y otro a memoria.
Solución: TLB (Translation Lookaside Buffer), una especie de caché de traducciones que incluye la MMU internamente.
- **Memoria:** las tablas de páginas son muy grandes y hay una por cada proceso activo. Si se usa todo el espacio de direccionamiento, la tabla tendría tantas entradas como páginas haya en el espacio lógico, aunque muchas estén marcadas como inválidas (apartado siguiente).

2.2.5 Tamaño de la Tabla de Páginas

Un problema que tiene la paginación es el tamaño de la tabla de páginas. Veamos esto con un ejemplo:

- Tenemos una **arquitectura de 32 bits** (4B).
- El **tamaño de página** es de **4KB** (2^{12} bytes).
- El **tamaño del campo de desplazamiento** es de **12 bits** (de los 32 bits, los 12 últimos) son para direccionar dentro de una página; es decir, que usamos 12 bits para **localizar el byte dentro de una página**.
- El **tamaño del número de página virtual** es de 20 bits, los 32-12 bits restantes, que se dedican al **número de página dentro del proceso**.

Esto quiere decir que un proceso puede tener 2^{20} páginas virtuales = 1 048 576 páginas virtuales; con lo que la tabla de páginas tendría 2^{20} entradas.

Esto permite que tengamos programas muy grandes en ejecución.

El problema es tener 2^{20} entradas en una tabla de página para un solo proceso, cuando no se están usando ni la mitad de las páginas, a lo mejor 1000, 2000 páginas.

Solución: paginación multinivel.

2.2.6 Paginación multinivel

Esta solución opta por “paginar las tablas de páginas”. La partición de la tabla de páginas permite al SO dejar particiones no usadas sin cargar hasta que el proceso las necesita. Aquellas porciones del espacio de direcciones que no se usan no necesitan tener tabla de página.

Paginación a dos niveles

Pretende que de los 20 bits que se le dedican al número de páginas (n), se dividan en dos campos:

- **p1**: los bits de la izquierda se referirán a una paginación a primer nivel, es decir, es el número de página (k);
- **p2**: los otros serán referentes a una paginación a un segundo nivel, que indican el desplazamiento de página ($n-k$).

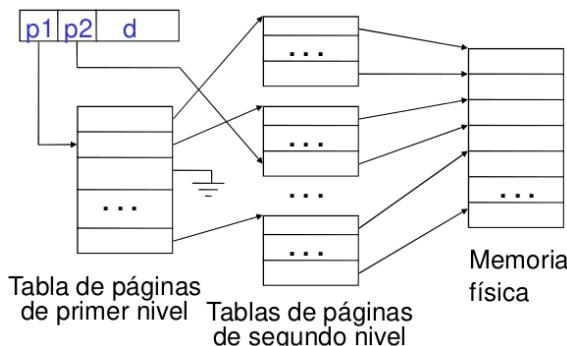
De esta forma, la dirección lógica quedaría:

- **p1**: bits de número de página;
- **p2**: desplazamiento de página;
- **d**: 12 bits del desplazamiento dentro de la página.



El esquema sería el siguiente:

- La **tabla de páginas de primer nivel** indican en **qué marco** de memoria principal se encuentra la **tabla de páginas de 2º nivel** que contiene la página que queremos.
- La **tabla de páginas de segundo nivel** indica en **qué marco** de la memoria principal está la **página cargada**. Las tablas de páginas de segundo nivel forman la tabla de páginas, solo que están divididas.
- **p1**: indica la entrada en la **tabla de páginas de primer nivel**, en esta entrada obtenemos la tabla de páginas de segundo nivel en la que se encuentra la página.
- **p2**: indica la entrada en la **tabla de páginas de segundo nivel**, en esta entrada obtenemos el marco físico en el que se encuentra la página en memoria principal.

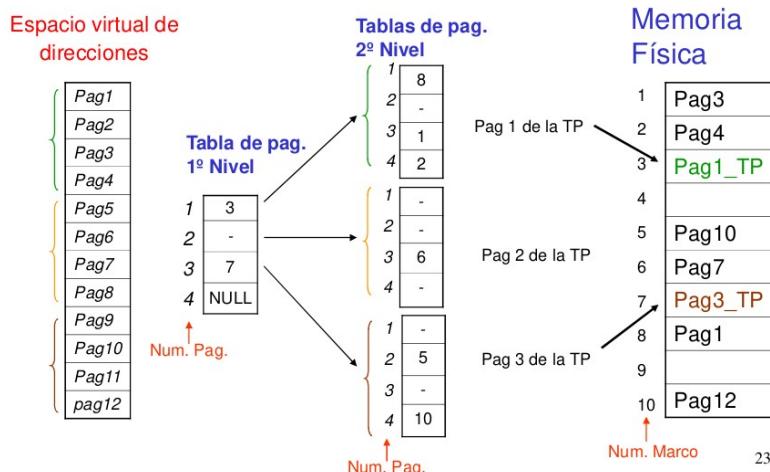


Por tanto, si una tabla de páginas de segundo nivel no tiene ninguna página cargada en memoria principal, no tendrá entrada en la tabla de páginas de primer nivel, con lo que dicha tabla de segundo nivel no tiene por qué estar cargada en memoria principal.

De esta forma, se consigue **liberar memoria y minimizar el tamaño de tabla de páginas**.

Un ejemplo es el siguiente:

- Un segmento de un proceso está dividido en 12 páginas.
- Suponemos que p_1 y p_2 son cada uno de 2 bits, con lo que el segmento del proceso podría tener $2^4 = 16$ páginas.
- Con 2 bits para p_1 , tenemos que el tamaño de la tabla de página de 1er nivel es de 4 entradas



(2²). Como el segmento tiene 12 páginas, y podemos llegar a 16, las últimas 4 están vacías, por ello la entrada 4 está a NULL.

Por tanto, las tablas de páginas de 2º nivel son 3, en vez de 4, ya que la última no se usaría, con lo que se estaría ajustando la tabla de páginas a lo que necesita el proceso actual.

- Con 2 bits para p_2 , tenemos que el tamaño de la tabla de páginas de 2º nivel es de 4 entradas.
- Se estaría dedicando 4 bits para las tablas de páginas, cuando en una arquitectura de 32 bits se le suele dedicar 20 bits, por lo que se realiza una importante reducción en el tamaño.

En la imagen se puede observar que:

- **En la entrada 1**, “página 1 de la TP”, hay un 3 que indica que en el marco 3 de memoria física se encuentra la tabla de páginas de 2º nivel. En este marco 3 encontramos la primera tabla de las tablas de páginas de 2º nivel.

Una vez tenemos la tabla de páginas de 2º nivel, vemos que tiene 4 entradas, una por página, que nos muestra en qué marco físico se encuentra la página. La página 1 en el marco 8, la 2 no está cargada, la 3 en el marco 1 y la 4 en el marco 2.

- **En la entrada 2**, “página 2 de la TP”, hay un “-” que indica que no hay tabla de páginas de 2º nivel cargada en memoria principal (se explica abajo).
- **En la entrada 3**, “página 3 de la TP”, hay un 7 que indica que en el marco 7 de memoria física se encuentra la tercera tabla de páginas de 2º nivel.

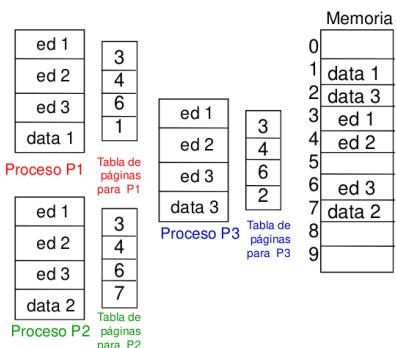
Una vez tenemos la tabla de páginas de 2º nivel, vemos que tiene 4 entradas. En las entradas 1 y 3 de la tabla (se corresponde con las páginas 9 y 11) tienen un guión, indicando que no están en memoria principal, mientras que la 2 (página 10) está en el marco 5 y la 4 (página 12) está en el marco 10.

¿Por qué en la entrada 2 de la tabla de páginas del 1er nivel hay un guión (-)?

Indica que la entrada en principio es válida, es decir, hay una tabla de páginas de 2º nivel existente en el proceso, pero ninguna de las páginas que se encuentran en la tabla de 2º nivel realmente están cargadas en memoria principal. Por tanto, tampoco está cargada dicha tabla de 2º nivel en memoria principal.

¿Qué pasaría si un proceso tuviera 11 páginas?

La tercera tabla del 2º nivel tendría su última entrada no asignada. Al sistema le resulta más sencillo trabajar con tablas de tamaño fijo ya que simplifican el trabajo y son más eficientes que ir ajustando el tamaño e ir modificando la reserva de memoria.



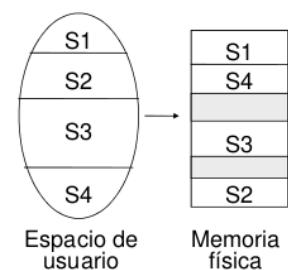
2.2.7 Páginas compartidas

Una copia de código de solo lectura compartido entre varios procesos.
Ej. editores, compiladores, sistemas de ventanas.

2.3 Segmentación

La **segmentación** es una técnica de gestión de memoria que pretende acercarse más al punto de vista del usuario.

- Un **programa** es una colección de unidades lógicas (**segmentos**) de tamaño variable, p. ej. procedimientos, funciones, pila, tabla de símbolos, matrices, etc.
- La **segmentación** de un programa la realiza el **compilador**.
- Cada dirección lógica se expresa mediante dos valores: número de segmento (s) y desplazamiento dentro del segmento (d).
- Dado que cada segmento se almacena de forma contigua y tienen longitud variable, este esquema presenta fragmentación externa.
- Permite a los programas y datos dividirse en espacios de direcciones lógicamente independientes, ayudando a la compartición y la protección.
- La memoria física se direcciona literalmente, por lo que hay que transformar cada dirección lógica en una dirección real. Esto se realiza con la tabla de segmentos.



La diferencia entre un segmento y una página es que las páginas dividen a un proceso en partes iguales, mientras que los segmentos no tienen por qué dividirlos en partes iguales.

PANDURA

MOTIVACIÓN



CONCENTRACIÓN



MEMORIA

**PANDURA**

Realiza tu diseño sobre el fondo de color negro.



*"Sin Pandora,
juegas en
desventaja."*



*"Memorizo un
50% más en el
mismo tiempo."*



MELENA DE LEÓN



BACOPA MONNIERI



CAMELLIA SINENSIS



GINGKO BILOBA



HUPERZIA SERRATA



COFFEA ARABICA

2.3.1 Tabla de Segmentos

Una **dirección lógica** se compone de:

- **s**: es el número de segmento;
- **d**: es el desplazamiento dentro de ese segmento.

s	d
----------	----------

Una **dirección física** se compone de:

- **s'**: es la dirección física donde reside el inicio del segmento en memoria;
- **d**: es el desplazamiento dentro de ese segmento.

s'	d
-----------	----------

La entrada de una tabla de segmentos contiene lo siguiente:

Número de segmento ↓	base	tamaño	presencia	modif.	protección
S'					

- **base**: dirección física donde reside el inicio del segmento en memoria;
- **tamaño**: longitud del segmento (los segmentos tienen longitud variable, por ello es necesario este campo).
- Presencia, modificación y protección es lo mismo que para las tablas de páginas.

La **tabla de segmentos** se mantiene en **memoria principal** y se le asocian dos registros:

- **Registro Base de la Tabla de Segmentos (RBTS)**: apunta a la tabla de segmentos (suele almacenarse en el PCB del proceso) [equivalente a la RBTP de paginación];
- **Registro Longitud de la Tabla de Segmentos (STLR)** indica el número de segmentos del proceso.
El nº de segmento **s**, generado en una dirección lógica, es legal si $s < STLR$ (suele almacenarse en el PCB del proceso), es decir, si el segmento es menor que el número de segmentos.

El SO mantiene una tabla de segmentos por cada proceso y en cada cambio de proceso irá informando a la MMU de qué tabla debe usar. Además, el SO también conoce qué partes de la memoria principal estarán libres y cuáles ocupadas.

2.3.3 Esquema de traducción

El **esquema de traducción** sería el siguiente:

- El proceso referencia la **dirección lógica** generada por la CPU, que es **s:d**, es decir, el número de segmento del proceso y el desplazamiento.

s	d
----------	----------

- Se busca en la tabla de segmentos el segmento **s**. La tabla de segmentos asociada a cada proceso tiene la siguiente estructura:

WUOLAH

	base	tamaño	presencia	modif.	protección
Número de segmento	S'	t			

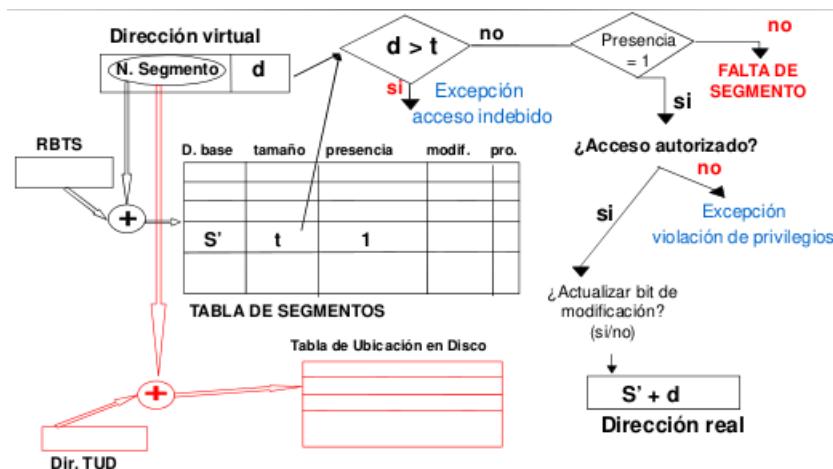
Como se puede estar ejecutando más de un proceso a la vez, habrá varias tablas de segmentos, una por proceso, por ello el acceso a la tabla de segmentos de un proceso se realiza con ayuda del **registro base de la tabla de segmento** (RBTS) asociado a cada programa.

Este RBTS se almacena en el PCB de cada proceso.

- Se comprueba el desplazamiento de la dirección lógica:
 - $d > t$: si el desplazamiento es mayor que el tamaño del segmento, ocurre una **excepción** de acceso indebido, ya que se estaría saliendo del propio segmento;
 - $d < t$: si el desplazamiento es menor que el tamaño de segmento, se mira el bit de presencia.
- Si el bit de presencia es:
 - 0 → ocurre un **fallo de segmento**, con lo que hay que traer el segmento del disco a memoria principal y actualizar los datos en la tabla de segmentos del proceso;
 - 1 → el segmento está en memoria principal, pero primero hay que comprobar que el proceso tiene permiso para acceder a dicho segmento.
 - No tiene permiso → ocurre una **excepción**, una violación de los privilegios del segmento.
- Si tiene permiso, ya tenemos la **dirección física** $s':d$, donde s' es la dirección física donde comienza el segmento, y d es el desplazamiento dentro de dicho segmento.



- Al tener ya la **dirección física**, se accede a la dirección en memoria principal y se desplaza d en dicho segmento para obtener el dato que el proceso necesita.



28

2.4 Segmentación paginada

- El espacio de direcciones físicas de un proceso puede ser **no contiguo**.
Un proceso se divide en varios segmentos, y dichos segmentos se dividen en varias páginas. Estas páginas que conforman los segmentos pueden no estar de forma continua en la memoria, sino en sitios diferentes.
- En una **segmentación paginada**, se tendrá un tabla de segmentos por proceso, y una tabla de páginas por segmento.

La segmentación:

- ✓ Proporciona soporte directo a las regiones de proceso.
- ✗ El problema es la variabilidad del tamaño de los segmentos y el requisito de memoria contigua dentro de un segmento complica la gestión de MP y MS.

La paginación:

- ✓ Permite un mejor aprovechamiento de la memoria y una base para construir un esquema de memoria virtual.
- ✗ El problema es que complica más los temas de compartición y protección (éstos van mejor en segmentación).

Por tanto, algunos sistemas combinan ambos enfoques, obteniendo la mayoría de las ventajas de la segmentación y eliminando los problemas de una gestión de memoria compleja.

En una segmentación paginada a dos niveles, primero se aplica la segmentación, es decir, se divide el proceso en segmentos, y después a cada segmento se le asigna la parte que le corresponde de la tabla de páginas.

La dirección lógica es la siguiente:

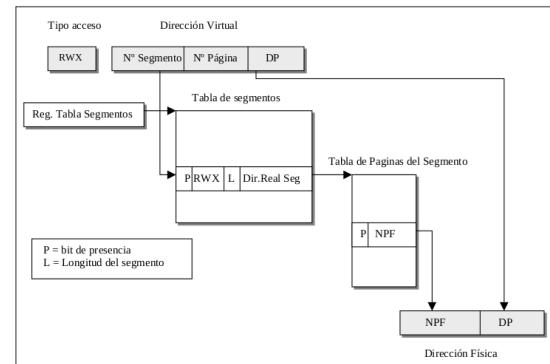
s p d'

- **s**: el número de segmento. El desplazamiento de un segmento se compone de las dos entradas siguientes, **d = p + d'**;
- **p**: el número de página;
- **d'**: el desplazamiento dentro de la página.

2.4.1 Esquema de traducción

El esquema de traducción es una mezcla de los dos anteriores:

- Se busca el segmento **s** en la tabla de segmentos con la ayuda del RBTS.
 - Se comprueba el tamaño y el desplazamiento **d**.
- La tabla de segmentos da la dirección base de la tabla de páginas del segmento **s**.
- En la tabla de páginas, se va a la entrada **p**, que proporciona el marco físico en el que se encuentra la página **p** del segmento **s**.
- Una vez se tiene la dirección física **m:d**, se accede al marco en memoria física y se desplaza **d**.



3. Gestión de la Memoria Virtual

3.1 Introducción

La gestión de memoria virtual la realiza el núcleo del sistema. Los sistemas suelen tener paginación a dos niveles. Hay varios criterios de clasificación respecto a:

- **Políticas de asignación:** de marcos de páginas en memoria principal a procesos; puede ser:
 - fija;
 - variable.
- **Políticas de sustitución:**
 - sustitución local,
 - sustitución global.
- **Políticas de búsqueda:**
 - **Paginación por demanda:** cuando un proceso necesita un marco de página, el sistema se pone en marcha como consecuencia de esa petición;
 - **Paginación anticipada:** puede ser que el sistema sea previsor, y antes de que se realice dicha petición de página, el sistema la cargue en memoria principal. Esto es sencillo, un sistema puede anticipar esto por la localización espacial; cuando se está ejecutando un código, lo más frecuente es que las siguientes líneas de código o las anteriores vuelvan a ser referenciadas.

Independientemente de la política de sustitución utilizada, existen ciertos criterios que siempre deben cumplirse:

- **Páginas "limpias" frente a "sucias":** se pretende minimizar el coste de transferencia. El bit de modificado sucio se encarga de que si una página está en disco y se lleva a memoria principal, y a la hora de volver a llevarla al disco no ha sufrido ninguna modificación con respecto a la copia en disco, no es necesario su transferencia a disco.
- **Páginas compartidas:** se pretende reducir el n.º de faltas de página.
- **Páginas especiales:** algunos marcos pueden estar bloqueados (ejemplo: buferes de E/S mientras se realiza una transferencia).

3.1.1 Políticas de asignación

Política de asignación fija

- Esta política proporciona un **número fijo** de marcos a todos los procesos. Cuando un proceso se ejecuta el número de marcos no varía, es fijo durante su ejecución.
- Este número se decide en el instante de creación del proceso y puede ser **por igual** o **según el tamaño**.

Asignación por igual

- Se asignan el mismo número de marcos a todos los procesos.
- Si hay m marcos, y n procesos. A cada proceso se le asignan m/n marcos.
Si tenemos 4 marcos y hay 2 procesos, se asigna a cada proceso 2 marcos.

PANDURA

MOTIVACIÓN



CONCENTRACIÓN



MEMORIA

**PANDURA**

Realiza tu diseño sobre el fondo de color negro.



*"Sin Pandora,
juegas en
desventaja."*



*"Memorizo un
50% más en el
mismo tiempo."*



MELENA DE LEÓN



BACOPA MONNIERI



CAMELLIA SINENSIS



GINGKO BILOBA



HUPERZIA SERRATA



COFFEE ARABICA

Asignación según tamaño

- El número de marcos que se asigna a un proceso es proporcional al tamaño del proceso.
- Esto se calcula con la siguiente fórmula:

$$a_i = (s_i / S) * m$$

- donde:
 - a_i es el número de marcos asociados al proceso i ;
 - s_i es el tamaño del proceso en número de páginas;
 - S es el número total de páginas que requieren todos los procesos en ejecución, se calcula como: $S = \sum s_i$
 - m es el número total de marcos de página que hay en memoria principal.

Política de asignación variable

- Esta política proporciona un **número variable de marcos por proceso**, va cambiando durante el tiempo de vida del proceso.
- Si un proceso, está causando **bastantes fallos de página de forma continua**, se le otorgarán marcos de página adicionales para reducir esta tasa de fallos.
- Si un proceso tiene pocos fallos de página, se reducirán los marcos reservados.
- Las dificultades de esta estrategia se deben a que el sistema operativo debe saber **cuál es el comportamiento del proceso activo**.
 - Lo que requiere una **sobrecarga software** por parte del sistema operativo y depende de los **mecanismos hardware** proporcionados por la propia plataforma del procesador.

3.1.1 Políticas de sustitución (o de reemplazo)

Cuando una página se pretende cargar en memoria principal pero no hay marcos de páginas libres, (cuando se produce un fallo de página) hay que sustituirla por una página que ya esté cargada en memoria. Se puede realizar dos sustituciones: local y global.

Política de reemplazo local

- Selecciona únicamente entre las páginas residentes del proceso que ha generado el fallo de página.
- Son más fáciles de analizar.

Política de reemplazo global

- Considera todas las páginas en la memoria principal que no se encuentren bloqueadas, aunque pertenezcan a otro proceso.
- Son fáciles de implementar con una sobrecarga mínima.

Hay una correlación entre el ámbito de reemplazo y el tamaño del conjunto residente.

	Reemplazo Local	Reemplazo Global
Asignación Fija	<ul style="list-style-type: none"> • El número de marcos asignados a un proceso es fijo. • Las páginas que se van a reemplazar se eligen entre los marcos asignados al proceso. 	<ul style="list-style-type: none"> • No es posible
Asignación Variable	<ul style="list-style-type: none"> • El número de marcos asignados a un proceso pueden variarse de cuando en cuando. • Las páginas que se van a reemplazar se eligen entre los marcos asignados al proceso. 	<ul style="list-style-type: none"> • Las páginas que se van a reemplazar se eligen entre todos los marcos de la memoria principal esto hace que el tamaño del conjunto residente de los procesos varíe.

Un **conjunto residente fijo** implica automáticamente una política de **reemplazo local**, ya que para reemplazar una página, se debe eliminar otra de la memoria principal del mismo proceso (ya que es un tamaño fijo).

Una **política de asignación variable** puede emplear la política de **reemplazo global**, ya que la sustitución de una página de un proceso en la memoria principal provoca que aumente su asignación en una página y disminuya en una en la del otro proceso del que se ha cogido la página.

Por tanto, hay 3 combinaciones posibles: **asignación fija y ámbito local**, **asignación variable y ámbito local** y **asignación variable y ámbito global**.

Asignación fija, ámbito local

- Un proceso en ejecución en la memoria principal tiene asignado un **número fijo** de marcos.
- Si se da un fallo de página, el SO debe elegir una página entre las residentes del proceso actual para realizar el reemplazo.
- Se utilizará los **algoritmos** de reemplazo del siguiente punto (3.2).

Desventajas:

- ✗ Si las **reservas** son demasiado **pequeñas**, va a haber demasiados fallos de página, haciendo que el sistema se **ralentice**.
- ✗ Si las **reservas** son demasiado **grandes**, habrá pocos programas en la MP y **tiempo perdido** de **swapping** o mucho **tiempo ocioso** del procesador.

Asignación variable, ámbito global

- Es la más **sencilla de implementar** y ha sido adoptada por un gran número de SOs.
- Cada uno de los procesos en la memoria principal tiene una serie de marcos asignados; y el SO mantiene una lista de marcos libres.
- Cuando hay un fallo de página, se añade un marco libre al conjunto residente del proceso y se trae la página a dicho marco.
- De esta forma, un proceso que sufre fallos de página **crecerá en tamaño**, lo que debería de **reducir** la tasa de fallos de página.
- No existe **ninguna disciplina** para indicar **qué proceso** debe perder una página del conjunto residente, por ello, el proceso que sufre la reducción no tiene por qué ser el óptimo.

Asignación variable, ámbito local

Esta asignación variable de ámbito local intenta resolver los problemas de la estrategia de ámbito global.

- Cuando se carga un proceso, se le asigna un cierto número de marcos de página;
- Si ocurre un fallo de página, se seleccionará otra página para realizar el reemplazo entre las páginas del conjunto residente de dicho proceso;
- De vez en cuando, se reevaluará la asignación proporcionada a cada proceso, incrementándose o reduciéndose para mejorar al rendimiento.
- Las decisiones relativas a aumentar o disminuir el tamaño del conjunto residente se toman de forma más **meditada** y se harán **contando con los indicios** sobre posibles demandas futuras de los procesos que se encuentran activos.
- Debido a la forma en la que se realiza esta valoración, esta **estrategia** es mucho más **compleja** que la política de reemplazo global simple, pero puede llevar a un **mejor rendimiento**.

3.1.3 Políticas de búsqueda (o de recuperación)

Política de recuperación con paginación bajo demanda

Con esta política, una página se trae a memoria sólo cuando se hace referencia a una posición en dicha página.

Si el resto de elementos en la política de gestión de la memoria funcionan correctamente, ocurriría lo siguiente:

- Cuando un proceso se arranca inicialmente, va a haber una ráfaga de fallos de página.
- Según se van trayendo páginas a la memoria, el principio de proximidad sugiere que las futuras referencias se encontrarán en las páginas recientemente traídas.
- Así, después de un tiempo, la situación se estabilizará y el número de fallos de página caerá hasta un nivel muy bajo.

Política de recuperación con paginación adelantada (prepaging)

Con esta política, se traen a memoria también otras páginas, diferentes de la que ha causado el fallo de página.

- Tiene en cuenta las **características** que tienen la mayoría de dispositivos de MS.
- Si las páginas de un proceso están almacenadas **de forma contigua** en MS, es mucho más **eficiente** traer a la memoria un número de páginas contiguas de una vez, en lugar de una a una durante un periodo de tiempo amplio.
- Es **ineficiente** si la mayoría de las páginas que se han traído **no se referencian a posteriori**.
- La política de paginación adelantada puede **emplearse**:
 - cuando **el proceso se arranca**, en cuyo caso el programador tiene que designar de alguna forma las páginas necesarias
 - cada vez que **ocurra un fallo de página**.

Este último caso es el más apropiado porque resulta completamente invisible al programador. Sin embargo, la completa utilidad de la paginación adelantada no se encuentra reconocida.

La paginación adelantada **no se debe confundir con el swapping**. Cuando un proceso se saca de la memoria y se le coloca en estado suspendido, todas sus páginas residentes se expulsan de la memoria. Cuando el proceso se recupera, todas las páginas que estaban previamente en la memoria principal retornan a ella.

Ventajas de la paginación anticipada

- ✓ Se puede optimizar el tiempo de respuesta para un proceso pero los algoritmos son más complejos y se consumen más recursos.

Desventajas de la paginación anticipada

- ✗ Puede equivocarse y cargar en memoria principal páginas que al final no se van a utilizar, ya sean por saltos en el código o cualquier motivo.
- ✗ Los algoritmos para la paginación anticipada son más costosos, complicados y consumen más que los de demanda.

Ventajas de la paginación por demanda:

- ✓ Se garantiza que solo están en MP las páginas necesarias en cada momento.
- ✓ La sobrecarga de decidir qué páginas llevar a MP es mínima.
- ✓ Se ha demostrado que es más interesante la paginación por demanda.

Rendimiento de la paginación por demanda

Al ser por paginación y tener gestión de memoria virtual, la paginación por demanda tiene una **sobrecarga** para el sistema. Esta sobrecarga está calculada como **TAE** (tasa de acceso efectivo a memoria).

$$\text{TAE} = (1 - p) * \text{acceso_a_memoria} + p * (\text{sobrecarga_falta_de_página} + [\text{sacar_fuera_una_página}] + \text{traer_la_página} + \text{sobrecarga_de_rearranque}).$$

Siendo p la tasa de falta de página ($p=0$ no hay faltas de páginas, $p=1$ toda referencia es una falta):

- **(1 - p) * acceso_memoria:** la probabilidad de que un proceso refiera a una posición de memoria definitiva y que se encuentre en memoria;
- **p*(sobrecarga_falta_de_página + [sacar_fuera_una_página] + traer_la_página + sobrecarga_de_rearranque):** en el resto de la probabilidad se produce una falta de página, por lo que hay que traerla al disco. Esta sobrecarga incluye:
 - **sobrecarga_falta_de_página:** gestionar la falta de página, en la que el proceso ha dirigido una dirección en la que la MMU dice que se produce una excepción de validez;
 - **sacar_fuera_una_pagina:** si no hay espacio, hay que gestionar la sustitución por una página que ya esté en memoria principal. Dependiendo del bit sucio, habrá que realizar la copia en disco de la página o no;
 - **traer_la_página:** hay que traer la página a memoria principal;
 - **sobrecarga_de_arranque:** ya que al haber una excepción, ha habido que ejecutar el manejador de validez.

La sobrecarga que implica los sistemas con paginación es habitualmente de un 20% más que si no tuviera implementada la paginación.

PANDURA

MOTIVACIÓN



CONCENTRACIÓN



MEMORIA



*"Sin Pandora,
juegas en
desventaja."*



*"Memorizo un
50% más en el
mismo tiempo."*



MELENA DE LEÓN



BACOPA MONNIERI



CAMELLIA SINENSIS



GINGKO BILOBA



HUPERZIA SERRATA



COFFEE ARABICA

Realiza tu diseño sobre el fondo de color negro.

3.1.4 Influencia del tamaño de página

- Cuanto más pequeñas sean las páginas:
 - el tamaño de las tablas de páginas es más grande,
 - el número de transferencias entre memoria principal y disco es mayor,
 - reducen la fragmentación interna.
- Cuanto más grandes sean las páginas:
 - grandes cantidades de información que no serán usadas están ocupando MP,
 - aumenta la fragmentación interna.
- Se busca un equilibrio en el tamaño de página, ni muy grandes ni muy pequeñas.

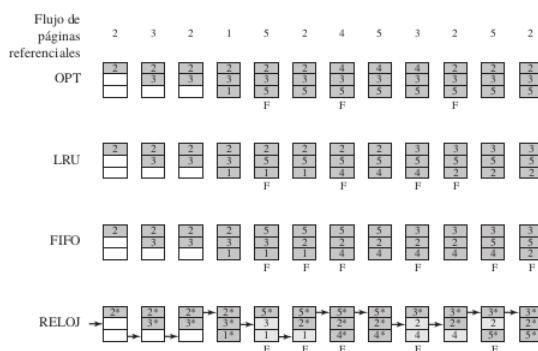
El tamaño de página siempre ha sido de 4KB, ya que no se desaprovecha mucho con la fragmentación interna. Hoy en día, el tamaño de página está aumentando a 16KB.

3.2 Algoritmos de sustitución

Veremos distintos algoritmos de sustitución y nos basaremos (por simplicidad) en que se utiliza una política de **asignación fija y sustitución local**. Estos algoritmos son independientes de la estrategia de gestión del conjunto residente.

Los algoritmos que se han desarrollado son:

- Óptimo
- FIFO (First In First Out)
- LRU (Last Recently Used)
- De reloj



La **cadena de referencia** es la secuencia de número de páginas que está referenciando el proceso durante su ejecución $\omega = r_1, r_2, r_3, \dots, r_i, \dots$

En los dibujos de los ejemplos tenemos que:

- Los números azules son la cadena de referencia.
- Las 4 filas son los 4 marcos de memoria principal que tiene asignado el proceso de forma fija para ir colocando las páginas que necesita.
- La primera vez que se utiliza una página siempre indica falta de validez ó falta de página, ya que inicialmente no hay cargada ninguna página en memoria principal, sólo el ejecutable.
- Con el asterisco se indica la falta de validez o falta de página.

3.2.1 Algoritmo Óptimo

Este algoritmo tomará como reemplazo aquella página que vaya a ser referenciada más tarde. Por ejemplo, si una página 1 se usará dentro de 10 instrucciones y otra página 2 será referenciada dentro de 5 instrucciones, se usará la página 1 como reemplazo.

En el ejemplo:

- **Falta de validez**, se carga 1.
- **Falta de validez**, se carga 2.
- **Falta de validez**, se carga 3.
- **Falta de validez**, se carga 4.
- **Referencia la 1**, que ya está en memoria principal.
- **Referencia la 2**, que ya está en memoria principal.
- **Falta de validez**, se carga 5, pero como todos los marcos están llenos, no hay ninguno libre, hay que sustituir la 5 por otra.

1,2,3,4,1,2,5,1,2,3,4,5
1 1 1 1 1 1 1 1 1 1 4 4
2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 5 5 5 5 5 5 5 5 5
* * * * * * *

Según el algoritmo óptimo, de las páginas cargadas se sustituye la que no se vaya a referenciar más o se referencie más tarde; es decir, después del 5, se referencian 1, 2, 3 y 4, por lo que 4 es la última en referenciarse.

Se sustituye la 5 por la 4.

- **Se referencian 1, 2 y 3** sin problema, ya que están en memoria principal.
 - **Falta de validez**, se carga 4, pero como no hay marcos libres, hay que sustituir. Como no sabemos cuál es la que no se va a referenciar más, sustituimos por la que lleve más tiempo, que en este caso es el 1.
 - Se sustituye la 4 por la 1.
 - **Referencia la 5**, que ya está en memoria principal.
- ◆ **Fallos de validez/páginas:** 6.

- x **Problemas:** es imposible de implementar, ya que es necesario que el SO tenga un conocimiento perfecto de los eventos futuros; es necesario tener un “conocimiento perfecto” de la cadena de referencia. Se puede estimar las referencias que va a realizar un proceso, pero se equivocará en muchos momentos. Además, que el sistema tenga que estimar es una sobrecarga, cuando puede que dicha estimación no sea correcta.
- ✓ **Se utiliza para medir cómo de bien se comportan otros algoritmos** para saber si son próximos al ideal.

3.2.2 Algoritmo FIFO

En este algoritmo, se reemplaza la página por la primera que ha llegado (la primera que ha entrado es la primera que sale), se sustituye la página por orden cronológico de llegada a MP.

Es sencilla de implementar.

- **Falta de validez x 4**, se carga 1, 2, 3 y 4. **1,2,3,4,1,2,5,1,2,3,4,5**
 - **Referencia la 1 y 2**, que ya están en memoria principal.
 - **Falta de validez**, se carga 5, pero como todos los marcos están llenos, no hay ninguno libre, hay que sustituir la 5 por otra.
Según el algoritmo FIFO, de las páginas cargadas se sustituye por la página que lleve más tiempo cargada.
Se sustituye la 5 por la 1.
 - **Falta de validez**, se sustituye 1 por otra. Se sustituye la 1 por la 2.
 - **Falta de validez**, se sustituye 2 por otra. Se sustituye la 2 por la 3 .
 - **Falta de validez**, se sustituye 3 por otra. Se sustituye la 3 por la 4.
 - **Falta de validez**, se sustituye 4 por otra. Se sustituye la 4 por la 1.
 - **Falta de validez**, se sustituye 5 por otra. Se sustituye la 5 por la 1.
- ◆ **Fallos de validez/páginas:** 10, vemos que no se acerca al óptimo.

1	1	1	1	1	5	5	5	5	4	4
2	2	2	2	2	1	1	1	1	1	5
3	3	3	3	3	2	2	2	2	2	2
4	4	4	4	4	4	3	3	3	3	3

✗ **Problemas:** sufre de la **Anomalía de Belady**: “más marcos no implican menos faltas de páginas”.

Anomalía de Belady

Por lógica al aumentar el número de marcos físicos en memoria, debería de reducir el número de fallos de página, sin embargo, la Anomalía de Belady (demostrado por Laszlo Belady) establece que es posible tener más fallos de páginas conforme aumentamos el número de marcos en la memoria física. **Esta Anomalía de Belady se presenta en este algoritmo.**

Con el ejemplo anterior de 4 marcos obtenemos 10 fallos de páginas. Si reducimos el número de marcos en memoria física, debería aumentar el número de fallos de página. Vamos a hacer el ejemplo de antes con 3 marcos:

	1	2	3	4	1	2	5	1	2	3	4	5
m 1	1	1	1	4	4	4	5	5	5	5	5	5
m 2		2	2	2	1	1	1	1	1	3	3	3
m 3			3	3	3	2	2	2	2	2	4	4
fp	x	x	x	x	x	x	x			x	x	

Con 3 marcos han ocurrido 9 fallos de página, menos que al tener 4. Esto se debe a la Anomalía de Belady, que especifica que más marcos de memoria física no implica menos fallos de páginas, como hemos visto.

La consecuencia de este descubrimiento fue la búsqueda de un algoritmo de reemplazo de páginas óptimo, el algoritmo anterior que hemos visto que es ideal y solo se utiliza para comparativas entre algoritmos.

3.2.3 Algoritmo LRU

Se sustituye la página que fue objeto de la referencia más antigua (Least Recently Used). Este algoritmo mira para atrás.

1,2,3,4,1,2,5,1,2,3,4,5

- **Falta de validez x 4**, se carga 1, 2, 3 y 4.
- **Referencia la 1 y la 2**, que ya está en memoria principal.
- **Falta de validez**, se carga 5, pero como todos los marcos están llenos, no hay ninguno libre, hay que sustituir la 5 por otra.

1	1	1	1	1	1	1	1	1	1	5
2	2	2	2	2	2	2	2	2	2	
3	3	3	3	5	5	5	5	4	4	
4	4	4	4	4	4	3	3	3	3	

Según el algoritmo LRU, de las páginas cargadas se sustituye por la que lleva más tiempo sin referenciar. Si miramos para atrás, la 3 es la que más tiempo lleva sin referenciarse. Se sustituye la 5 por la 3.

- **Se referencia la 1 y 2**, que ya están en memoria principal.
- **Falta de validez**, se sustituye 3 por otra. Se sustituye la 3 por la 4, que es la que más lleva sin ser referenciada.
- **Falta de validez**, se sustituye 4 por otra. Se sustituye la 3 por la 5, que es la que más lleva sin ser referenciada.
- **Falta de validez**, se sustituye 5 por otra. Se sustituye la 5 por la 4, que es la que más lleva sin ser referenciada.

- ◆ **Fallos de validez/páginas:** 8.

- x **Problema:** la implementación del algoritmo, con lo que hay un mayor coste.

Implementaciones de LRU

¿Cómo sabemos realmente cuál es la página cuya referencia es más antigua? Hay dos formas de implementar el algoritmo: con pila o con contadores.

Con pila

Las páginas que están cargadas en memoria principal se encuentran en la pila.

- Cuando una página es referenciada, se pasa a la base de la pila.
- Cuando hay que sustituir una página, se sustituye la que está en el tope de la pila.

Estas pilas se suelen montar con listas enlazadas, en las que hay que cambiar las posiciones de la pila con punteros. Parece interesante, pero la implementación es pesada, ya que el núcleo tiene que mover punteros, direcciones, actualizar la lista.

Con contador

- Cada entrada de la tabla de páginas tiene un contador. Cada vez que se referencia la página, se copia el tiempo del reloj en el contador.
- Cuando necesitamos cambiar una página, se miran todos los contadores y se elige la que tiene el menor tiempo.

Hay que realizar una búsqueda para saber cuál es la que tiene el menor tiempo. No es del todo ineficiente, pero requiere una búsqueda que requiere tiempo.

Solución: el algoritmo del reloj.

PANDURA

MOTIVACIÓN



CONCENTRACIÓN



MEMORIA

**PANDURA**

Realiza tu diseño sobre el fondo de color negro.



*"Sin Pandora,
juegas en
desventaja."*



*"Memorizo un
50% más en el
mismo tiempo."*



MELENA DE LEÓN



BACOPA MONNIERI



CAMELLIA SINENSIS



GINGKO BILOBA



HUPERZIA SERRATA



COFFEA ARABICA

3.2.4 Algoritmo del reloj

Se plantea como un algoritmo diferente porque es una aproximación al LRU.

Consiste en tener un puntero que va a recorrer todas las entradas de las tablas de páginas en el mismo sentido. Cuando llegue a la última entrada, vuelve a la primera, haciendo un recorrido de forma circular.

- Cada página tiene asociado un **bit de referencia R**, que indica si la página ha sido referenciada desde la última vez que la aguja del algoritmo pasó por ella.

La página tiene entonces el número de marco, el bit de presencia, el modificado y la protección y además se le añade el bit de referencia. Esto también hace que en la tabla de páginas haya una columna más que es el bit de referencia.

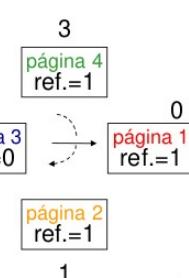
- Cuando una página llega a memoria se establece dicho bit a 1; en cada referencia el hardware lo pone a 1.
- Los marcos de página se representan por una lista circular y un puntero a la página visitada hace más tiempo.
- La selección de una página se realiza consultando el bit R del marco actual:
 - Si $R = 1$, se inicializa $R = 0$ y el puntero avanza al marco siguiente y vuelve a consultar.
 - Si $R = 0$, se selecciona para sustituir y se incrementa la posición.

Este algoritmo es **muy eficiente** a la hora de implementarlo, no hay que realizar cálculos complejos, solo la comparación de los valores del bit de referencia.

Es el algoritmo que se suele utilizar para sustituir páginas.

Ejemplo cuando hay que sustituir una página por otra:

- El puntero está apuntando a la página 1 en el marco 0, cuyo bit de referencia es 1 (indica que ya ha sido referenciada). Como lo acaba de referenciar, establece su referencia a 0.
- Avanza y referencia a la página 2 en el marco 1, cuyo bit de referencia es 1. Cambia su bit de referencia a 0.
- Avanza y referencia a la página 3 en el marco 2. Como su bit de referencia es 0, sustituye esta página por la nueva.



Tanto este como el LRU sustituyen la página que hace tiempo que no ha sido referenciada pero en este algoritmo del reloj no se garantiza que sustituye la más antigua referenciada.

Otra explicación

Cada página tiene asociado un bit de referencia R, denominado bit de referencia. Dicho bit se establece a 1 cuando:

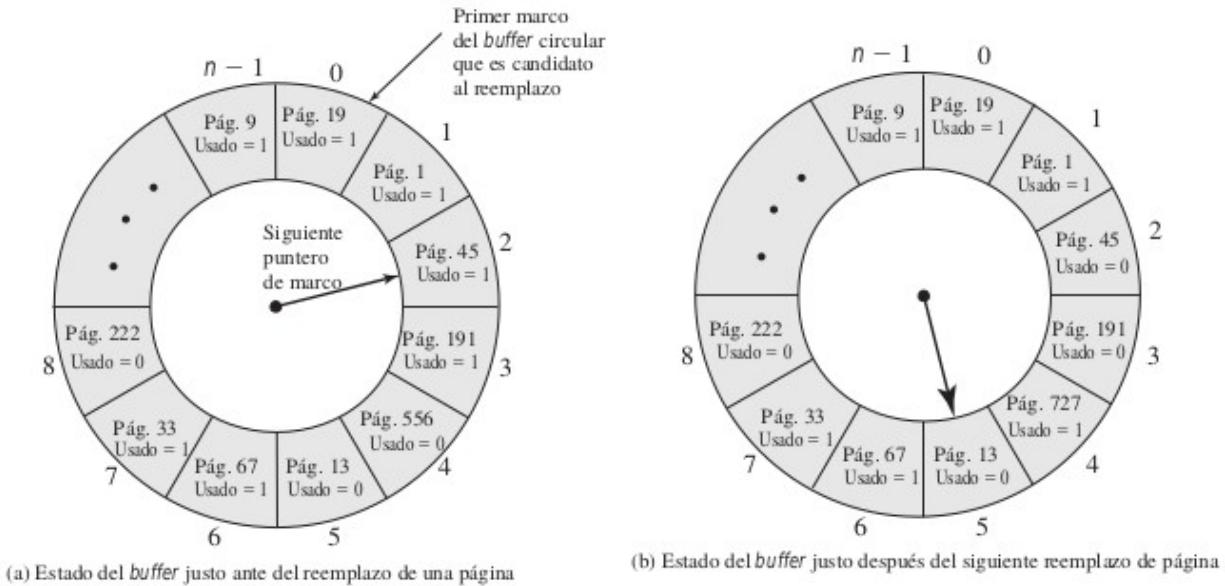
- se trae por primera vez a memoria;
- la página vuelve a utilizarse (después de la referencia generada con el fallo de página inicial);

Un ejemplo:

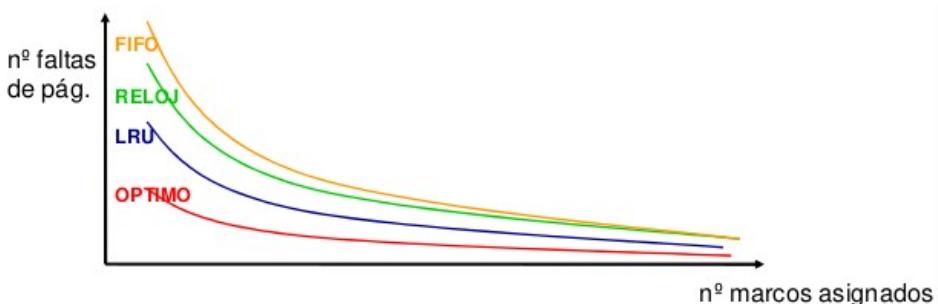
- Tenemos el siguiente buffer con las páginas candidatas para el reemplazo y con el puntero en la posición 2. Dicho puntero indica la siguiente página a la que acaba de ser actualizada.
- Se solicita la página 727.

WUOLAH

- En la posición 2, el bit de usado está a 1, se pone a 0 y se va al siguiente.
- En la posición 3, está a 1, se vuelve a poner a 0 y se va al siguiente.
- En la posición 4, el bit está a 0, por lo que se puede usar para realizar el reemplazo de página. El bit se pone a 1 y el puntero pasa a la siguiente posición.



3.2.5 Comparación



Cuando el número de marcos asignados es grande, los 4 algoritmos se asemejan, pero cuando el número de marcos asignados es pequeño:

- El óptimo produce menos fallos de páginas con diferencia.
- El segundo mejor es el LRU.
- Le sigue el de reloj, que aunque realice más fallos de páginas es más rápido en la implementación, muy eficiente.
- El FIFO es el que más fallos de página produce.

Hoy en día se usan algoritmos de reloj o variantes.

Conclusión:

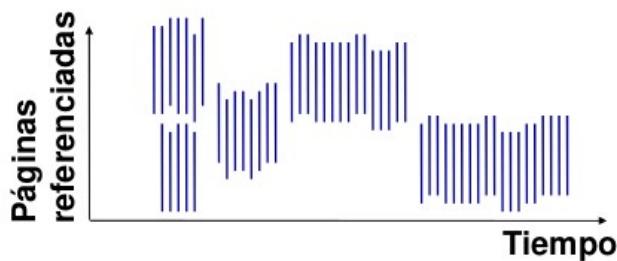
- Influye más la cantidad de memoria principal disponible que el algoritmo de sustitución usado

3.3 Comportamiento de los programas

Viene definido por la secuencia de referencias a página que realiza el proceso.

Importante para maximizar el rendimiento del sistema de memoria virtual (TLB, alg. Sustitución, ...).

La siguiente imagen es una representación de en un intervalo de tiempo razonable cuáles son las páginas que está referenciando un proceso:



3.3.1 Propiedad de localidad

La localidad se divide en dos tipos:

- **Temporal:** una posición de memoria referenciada recientemente tiene una probabilidad alta de ser referenciada en un futuro próximo (ciclos, rutinas, variables globales, ...). Como un bucle.
- **Espacial:** si cierta posición de memoria ha sido referenciada es altamente probable que las adyacentes también lo sean (array, ejecución secuencial, ...). Como las siguientes instrucciones de un código.

3.3.2 Conjunto de Trabajo

El **conjunto de trabajo** (Working Set) de un proceso es el conjunto de páginas que son referenciadas frecuentemente en un determinado intervalo de tiempo, es decir, teniendo en cuenta el **ancho de ventana**.

El ancho de ventana es el intervalo de tiempo comprendido entre t y $t-\Delta$:

$$WS(t,\Delta) = \text{páginas referenciadas en el intervalo de tiempo } (t - \Delta, t]$$

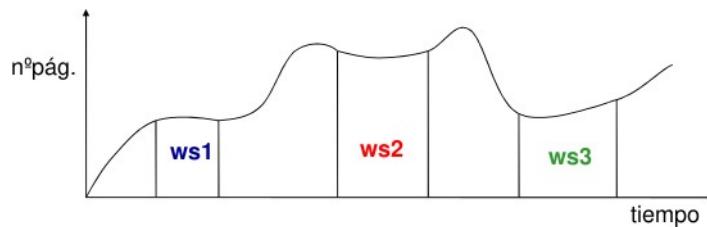
Por ejemplo, en las últimas 5 referencias del proceso a memoria principal, cuáles son las páginas que ha referenciado.

Observaciones:

- Mientras el conjunto de páginas necesarias puedan residir en memoria principal, el número de faltas de página no crece mucho.
- Si eliminamos de memoria principal páginas de ese conjunto, la activación de paginación crece mucho.

Propiedades

- Los conjuntos de trabajo son transitorios, puede aumentar o disminuir.
- No se puede predecir el tamaño futuro de un conjunto de trabajo.
- Difieren unos de otros sustancialmente.



3.3.3 Teoría del Conjunto de Trabajo

- Un proceso sólo puede ejecutarse si su conjunto de trabajo está en memoria principal.
- Una página no puede retirarse de memoria principal si está dentro del conjunto de trabajo del proceso en ejecución.

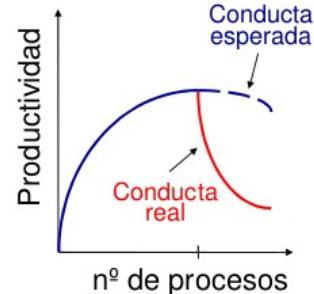
3.4 Hiperpaginación (Thrashing)

Si el número de marcos de página asignados a un proceso no es suficiente para almacenar las páginas referenciadas activamente por el mismo, se producirá un número elevado de fallos de página. A esta situación se le denomina **hiperpaginación** (thrashing).

Cuando se produce la **hiperpaginación**, el proceso pasa más tiempo en la cola de servicio del dispositivo de paginación, es decir, ejecutando algoritmos de sustitución que ejecutando el proceso en sí.

Se ha detectado que cuando los procesos no mantienen en memoria principal su conjunto de trabajo, es cuando se produce la hiperpaginación.

Dependiendo del tipo de asignación utilizado, este problema puede afectar a procesos individuales o a todo el sistema.



- **Asignación fija:**
 - si el número de marcos asignados a un proceso no es suficiente para albergar su conjunto de trabajo en una determinada fase de su ejecución → **hiperpaginación**;
 - provoca el aumento considerable de su tiempo de ejecución;
 - no afecta al resto de los procesos del sistema directamente.
- **Asignación dinámica:**
 - el número de marcos asignados a un proceso se va adaptando a sus necesidades, por lo que, no debería presentarse este problema;
 - pero si el número de marcos de página existentes en el sistema no son suficientes para almacenar los conjuntos de trabajo de todos los procesos → hiperpaginación;

¿BUSCAS LIBROS O EBOOKS SOBRE ENFERMERÍA?

AXON librería especializada en Ciencias de la Salud.

AXON
axon.es



Encuentra todos los libros y eBooks para tu especialización, además del material complementario que necesites.

- la utilización del procesador disminuye, debido al tiempo dedicado a los fallos de página;
- disminuye de una forma drástica debido al aumento del número de procesos (aumentando así la tasa de fallos de página) y al aumento del tiempo de servicio del dispositivo de paginación;
- **Solución:** suspender uno o varios procesos liberando sus páginas.

Formas de evitar la hiperpaginación

- **Algoritmos de asignación variables:** asegurar que cada proceso existente tenga asignado un espacio en relación a su comportamiento, es decir, intentar tener controlado el conjunto de trabajo, lo que se consigue haciendo que el número de marcos sea variable.
- **Algoritmos de regulación de carga:** actuar directamente sobre el grado de multiprogramación, no permitiendo que entren tantos procesos en memoria principal.

3.5 Algoritmos de asignación variable y sustitución global

3.5.1 Algoritmo basado en el modelo del WS

Para ajustar muy bien el conjunto de trabajo de los procesos con asignación variable, se realiza en base al conjunto de trabajo de dicho proceso. Por ello, es importante el ancho de ventana V , establecer bien dicho valor.

En cada momento t en que se hace referencia a una posición de memoria, se determina el conjunto de trabajo de la siguiente forma:

$WS = \text{páginas referenciadas en el intervalo } (t - V, t]$

solo estas páginas se mantienen en memoria principal.

3.5.2 Algoritmo Frecuencia de Falta de Página (FFP)

En este algoritmo hay un umbral establecido L y su definición es:

- se calcula el tiempo actual,
- y la diferencia con el tiempo de la última falta de página que produjo el mismo proceso.
- Si la diferencia es mayor que el umbral L , supone que el proceso tiene demasiados marcos de páginas, que no está produciendo muchas faltas de páginas con lo que se le puede quitar marcos de página.
- Si la diferencia es menor o igual que el umbral L , supone que el proceso está produciendo muchos fallos de páginas, su conjunto de trabajo debería ser más grande y tiene pocos marcos de páginas, con lo que el sistema debe asignarle a dicho proceso más marcos de páginas.

Este es más fácil de implementar que el algoritmo anterior, que se basa directamente en el conjunto de trabajo. Este algoritmo es el que se utiliza.