SISTEMAS OPERATIVOS AVANZADOS

Tema 1. Gestión de memoria

La memoria es un recurso utilizado por cualquier programa durante su ejecución, además, es un recurso que debe ser compartido por todos los procesos que se encuentren activos en el sistema. Por lo que es fundamental garantizar que se utilice de un modo eficiente y seguro, evitando que unas aplicaciones puedan interferir con otras y garantizando además que las aplicaciones puedan intercambiar datos cuando sea necesario.

El **espacio de memoria virtual** es un espacio independiente para cada uno de los procesos.

1. ABSTRACCIONES DE DIRECCIONAMIENTOS

El espacio de direccionamiento es un conjunto de direcciones referenciables.

- ESPACIO DE DIRECCIONAMIENTO VIRTUAL -> Independiente para cada proceso
- ESPACIO DE DIRECCIONAMIENTO FÍSICO -> repartido entre todos los procesos

Los procesos solo referencian direcciones virtuales.

Tiene que haber una traducción de dirección virtual a física transparente al proceso.

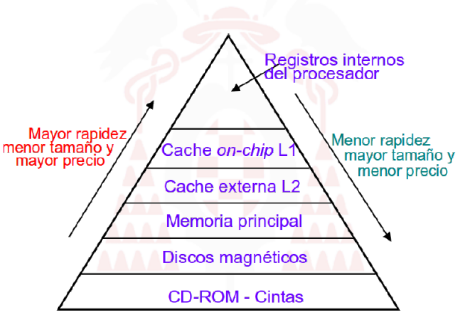
La CPU genera direcciones de memoria virtuales (en el caso del multiproceso) por lo que es necesario la MMU (unidad de gestión de memoria, que traduce esas direcciones de memoria virtuales en direcciones de memoria físicas).

2. JERARQUÍA DE MEMORIA

La jerarquización de la memoria es un intento de aumentar el rendimiento de los computadores. Para ello se aprovechan los avances tecnológicos en el diseño de memorias y la localidad de los programas.

- MEMORIAS RÁPIDAS: tienen un coste elevado y una capacidad pequeña
- MEMORIAS LENTAS: son baratas y tienen una capacidad alta

Para que esto funcione es necesario el principio de localidad de los programas.



3. PRINCIPIO DE LOCALIDAD

El principio de localidad es una propiedad empírica que se basa en que los procesos tienden a concentrar sus referencias en un intervalo de tiempo en un subconjunto de su espacio de direcciones.

Donnald Kuth decía que los programas, normalmente, tienen un perfil muy desigual, con unos pocos picos agudos. También encontraremos que al menos el 4% de un programa, generalmente, representa más de la mitad de su tiempo de ejecución.

Podremos decir que la localidad de referencias consiste en que los procesos tienden a concentrar sus referencias a memoria en un determinado intervalo de tiempo en un subconjunto bien definido de su espacio total de direcciones.

La localidad puede ser de dos tipos:

- *Localidad espacial*

Establece que si una posición de memoria es accedida en un determinado instante de tiempo t es muy probable que en un instante $t+\Delta t$ las posiciones de memoria cercanas también sean accedidas. Esto implica

que **una vez hecha una referencia a una dirección de memoria es muy probable que las direcciones de memoria cercanas también sean referenciadas**. En apoyo a esta observación encontramos:

- Ejecución secuencial del código
- Tendencia de los programadores a colocar próximas entre sí variables relacionadas
- Acceso a estructuras de datos de tipo matriz o pila

- *Localidad temporal*

Una vez hecha una referencia a una posición de memoria en un determinado instante *t*, es muy probable que esa misma posición de memoria sea accedida en un instante *t+Δt* cercano, dicho de otro modo, **las direcciones de memoria referenciadas recientemente tienen una alta probabilidad de ser referenciadas en un futuro próximo**. En apoyo a esta observación encontramos:

- Formación de ciclos (for, while…)
- Llamada a subrutinas
- Pilas

4. FRAGMENTACIÓN

Por fragmentación siempre debemos entender el **desaprovechamiento de la memoria**. En general, la fragmentación se usa como criterio para determinar la calidad de los distintos esquemas de gestión. Cuanto mayor sea la fragmentación, peor será el esquema de gestión de memoria empleado.

La fragmentación puede ser de dos tipos:

- Fragmentación interna -> **se debe a la diferencia de tamaño entre la partición de memoria y el objeto residente dentro de ella**, se produce cuando tenemos particiones de tamaño fijo.
  Supongamos que tenemos una memoria de 10 KB dividida en cinco regiones de 2KB cada una. En cada región se puede albergar un único trabajo de tamaño máximo 2 KB. Si recibimos un trabajo que requiere 1,5KB puede cargarse en cualquier bloque, pero siempre habrá una parte desaprovechada de 0.5 KB (pérdida conocida como fragmentación interna).

- Fragmentación externa **-> se debe al desaprovechamiento de memoria entre particiones como consecuencia de la dispersión del espacio libre en áreas discontinuas**. Se produce cuando tenemos bloques/segmentos de diferentes tamaños.
  Supongamos que en un instante determinado tenemos tres zonas libres de memoria no contiguas, con tamaños de 2 KB, 5KB y 7KB. Si en ese instante llega un trabajo de tamaño 13KB, este no se puede cargar porque, aunque hay espacio suficiente esta no es contigua con lo que la memoria quedará desaprovechada (fragmentación externa)

  NOTA: ES MEJOR TENER FRAGMENTACION INTERNA ya que no necesitamos más recursos, mientras que en la externa sí que los necesitamos.

La solución a ese problema es la **compartición de memoria**. La fragmentación es en MP, no tiene sentido halar de fragmentación en MV.

5. REUBICACION

La reubicación **es la posibilidad de ubicar a los programas en distintas zonas de memoria**. Las direcciones de memoria donde se carga un programa cuando lo ejecutamos repetidas veces no son siempre las mismas, puesto que el estado de memoria en un sistema multiprogramado varía de unos instantes a otros.

Dependiendo de si el programa se puede ubicar en diferentes zonas de memoria en el momento de carga o durante su ejecución, hablaremos de reubicación estática o dinámica respectivamente.

- *Reubicación estática*

Se realiza antes o durante la carga del programa. Los programas no pueden ser movidos una vez inicializados.

- *Reubicación dinámica*

Las direcciones de un programa antes de cargarlo en memoria, el cambio de direcciones se hace dinámicamente durante cada referencia, en tiempo de ejecución. Necesita hardware adicional (MMU) y los programas pueden moverse en tiempos de ejecución

6. PROTECCIÓN Y USO COMPARTIDO

Cuando tenemos varios programas cargados simultáneamente en memoria es necesario establecer unos mecanismos de protección con el fin de delimitar el acceso a memoria tanto por parte del sistema operativo como de los procesos de usuario.

Otro aspecto que es necesario tener en cuenta es la necesidad que tienen los procesos de intercambiar información. Este intercambio se realiza en última instancia por medio de zonas de memoria compartidas que el sistema operativo debe controlar. El mecanismo de compartición puede implicar en el caso más sencillo a dos aplicaciones, pero en el caso general pueden verse implicados un número indeterminado.

La protección de las páginas se logra añadiendo bits de acceso a las diferentes entradas de la tabla de páginas.

- *Métodos de protección*

Con objeto de garantizar la protección de zonas de memoria y garantizar una compartición controlada, el sistema operativo necesitará que el hardware le proporcione unos servicios más o menos evolucionados. Algunos mecanismos para este propósito son:

  - **REGISTROS LIMITES**: se establece cuáles son las direcciones mínima y máxima que un proceso puede generar.
  - **REGISTROS BASE Y LÍMITE**: Se compara el registro límite con la máxima dirección que puede generar el proceso y si esta es menor, se suma el valor del registro base.
  - **Bits de protección en memoria**: Cada proceso tiene asignado unos determinados bits de acceso en memoria. Cada bloque de memoria tiene también asignados unos bits que indican que procesos pueden acceder a este bloque.
  - **Tablas de acceso**: típicas de sistemas segmentados y paginados. En ellas se pueden establecer diferentes derechos de acceso y protección a cada zona de memoria.

7. EVOLUCIÓN HISTÓRICA DEL GESTOR DE MEMORIA

El gestor de memoria es la parte del sistema operativo encargado de asignar y liberar la memoria a los procesos, llevar la contabilidad de la misma y proporcionar los mecanismos básicos de protección y compartición.
A la hora de establecer si un gestor de memoria es mejor o no que otro suelen considerarse los siguientes aspectos: la fragmentación producida, la complejidad del gestor y los retardos introducidos en los accesos.

- **Máquina desnuda** -> el sistema no proporciona ningún servicio
- **Monitor monolítico** -> además del sistema operativo, hay un único proceso
- **Memoria particionada contigua**:

- **Multiprogramación con número fijo de tareas (MFT)**-> Particiones de tamaño fijo que se crean al arrancar el sistema
- **Multiprogramación con número variable de tareas (MVT)**-> las particiones son de tamaño variables y se crean cuando un proceso lo necesita

- **Memoria particionada no contigua**:
  El contenido de un proceso puede ser distribuido entre diferentes particiones separadas en memoria.
  La memoria está organizada en particiones:
  - De tamaño variable -> segmentos
  - De tamaño fijo -> marcos

  Posea **una tabla de descripción de particiones, la cual es independiente por proceso y se construye en tiempo de carga del proceso en memoria**.

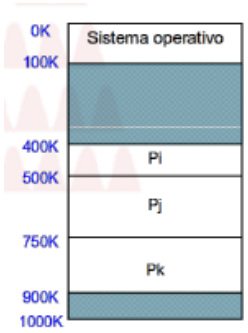
8. MEMORIA PARTICIONADA NO CONTIGUA

El contenido de un proceso puede ser distribuido entre diferentes particiones en memoria:

- De tamaño variable → **segmentos**.
- De tamaño fijo → **marcos**.

Tabla de descripción de particiones:

- **Independiente por proceso**.
- Se construye en tiempo de carga del proceso en memoria.

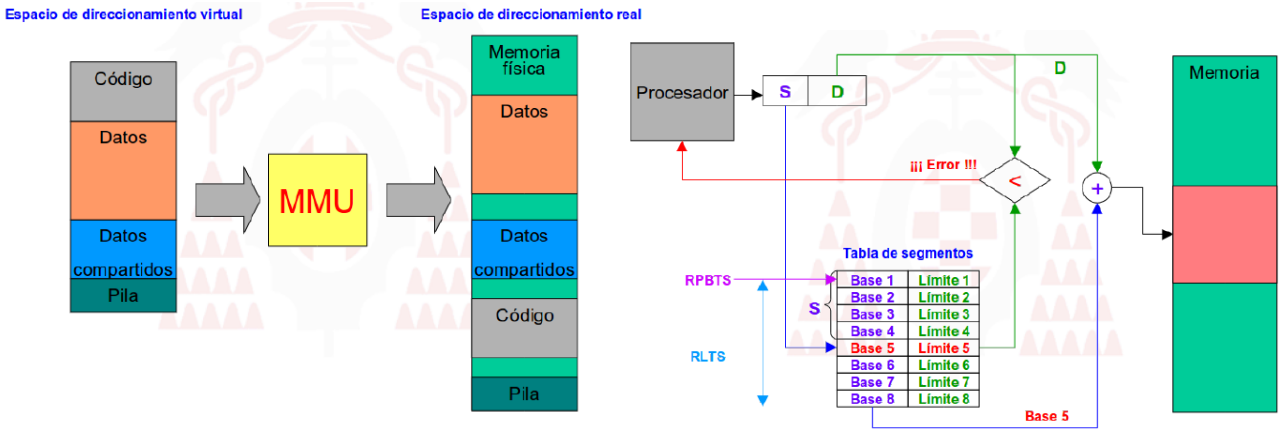


Número de la partición	Base de la partición	Tamaño de la partición	Estado de la partición
0	0K	100K	ASIGNADA
1	100K	300K	LIBRE
2	400K	100K	ASIGNADA
3	500K	250K	ASIGNADA
4	750K	150K	ASIGNADA
5	900K	100K	LIBRE

9. SEGMENTACIÓN

Inicialmente la memoria física está organizada como un único bloque vacío donde se crean particiones de tamaño variable (segmentos) a medida que se van necesitando. El espacio de direccionamiento virtual se organiza en segmentos.

- Posee mecanismo de protección y permite uso compartido.
- Las direcciones virtuales tienen dos componentes: nº de segmento y desplazamiento.
- Tabla de particiones denominada Tabla de Segmentos (**TDS**).
  - Si la TDS es muy grande, es necesario almacenarla en memoria principal apuntada por un registro (RPBTS) → necesarias dos referencias a memoria por acceso.



Ventajas:

- No produce fragmentación interna.
- Crecimiento dinámico de los segmentos (suministra reubicación dinámica).
- Proporciona tanto protección como uso compartido.
- **Los diferentes bloques de un proceso, código, datos, pila, área de datos compartidos, etc., pueden estar situados en áreas de memoria no contiguas, en segmentos diferentes**.

Inconvenientes:

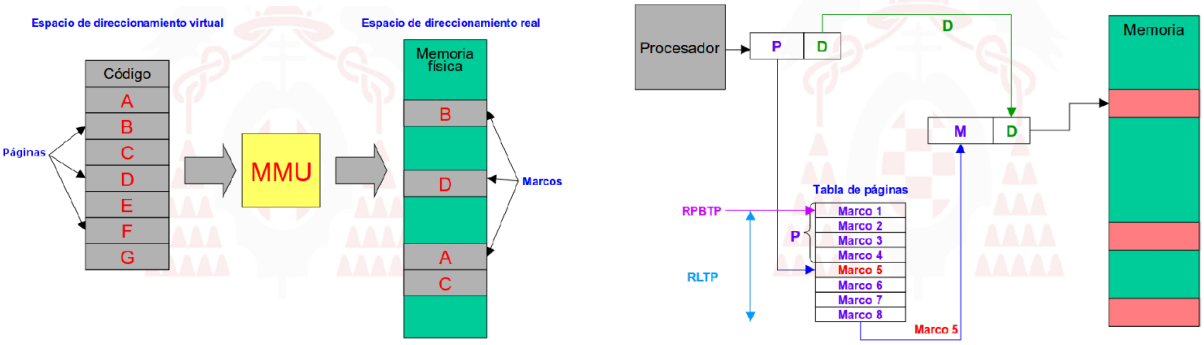
- Necesita compactación de memoria.
- Puede aparecer fragmentación externa.

10. Paginación (sistema de asignación de memoria no contigua)

Inicialmente la memoria física está organizada en particiones de tamaño fijo (marcos). El espacio de direcciones virtuales de un proceso está divido en bloques de tamaño fijo llamados páginas (estas se almacenan en memoria física en lo que se conoce como **marcos de página**).

Cada página virtual va a tener su marco de página físico asociado, por tanto, página y maro **tienen el mismo tamaño**.

- Las direcciones virtuales tienen dos componentes: **nº de página virtual** y **desplazamiento**.
- Posee mecanismo de protección y permite uso compartido.
- Tabla de particiones denominada Tabla de Mapa de Páginas (**TMP**).
    - Si la TMP es muy grande, es necesario almacenarla en memoria principal apuntada por un registro (RPBTP).



Ventajas:

- No produce fragmentación externa.

Inconvenientes:

- Puede aparecer fragmentación interna.

Con páginas grandes aumenta la fragmentación interna, pero disminuye el tamaño de la TMP y viceversa.

Si el nº de páginas es grande, la zona de memoria ocupada por la TMP puede ser excesiva, por lo que hay que paginar la tabla de páginas.

11. COMBINACIÓN DE MECANISMOS

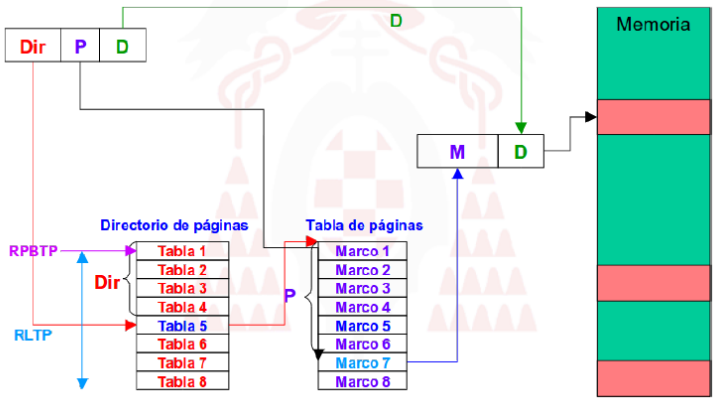
Es posible combinar los esquemas de paginación y segmentación. Se obtienen las ventajas de ambos esquemas a costa de complicar el hardware. Las posibles combinaciones son:

- Segmentación paginada.
- Paginación segmentada (no se emplea en la práctica).

12. PAGINACIÓN PAGINADA

La paginación paginada surge con objeto **de evitar tener que trabajar con tablas de páginas excesivamente grande**. Combinando la paginación paginada con un esquema de MV, sólo aquellas entradas de la tabla de páginas que se utilicen en cada momento se mantienen en MP, el resto no tiene por qué.

Las tablas de páginas se organizan en **directorios de página**. Cada directorio de página permite localizar determinadas entradas de la tabla de páginas, cada entrada de directorio permite localizar la dirección física donde comienza su tabla de páginas correspondiente.

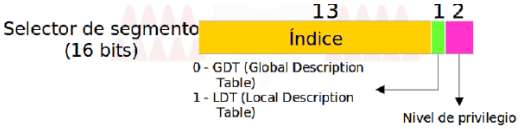


13. CASOS DE ESTUDIO: Gestión de memoria del Pentium.

MMU del Pentium:

EL Pentium soporta segmentación, paginación y segmentación paginada (la más habitual). La dirección lógica está compuesta por un selector de segmento (13+1 bits) y un desplazamiento (32 bits).

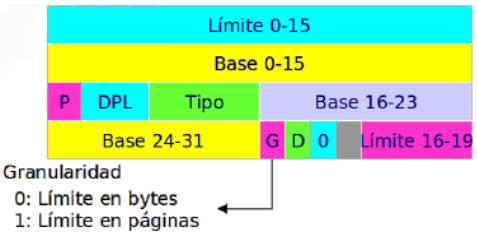
El selector de segmento es uno de los siguientes registros: CS, DS, ES, SS, FS, GS.



Formato del descriptor de segmento:

- LDT (Local Descriptor Table) → una por proceso.
- GDT (Global Descriptor Table) → una por sistema.
- Número máximo de entradas en cada tabla → 2^{13}.

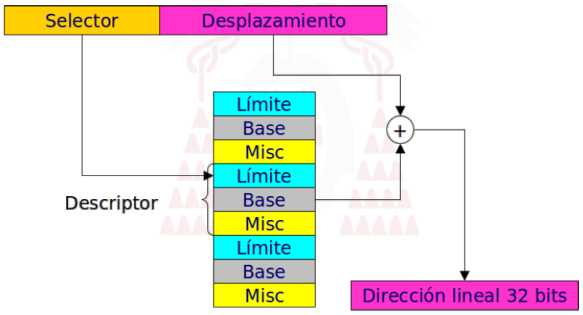
- Cada entrada en la tabla de segmentos se denomina descriptor.
- Tamaño del descriptor → 8 bytes.

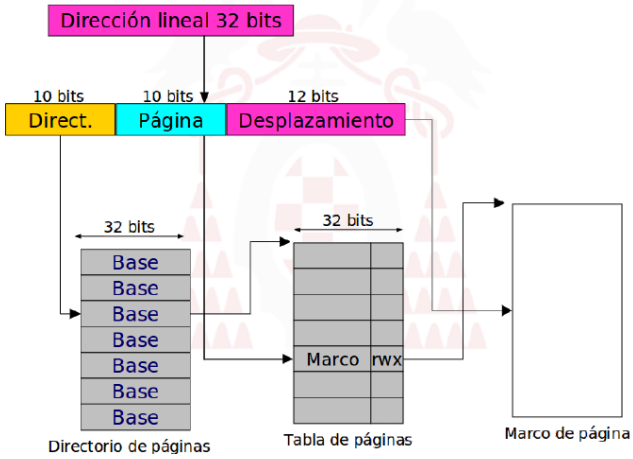


Descriptor de segmento:

- Dirección base (32 bits).
- Límite (20 bits).
- Atributos y privilegios (12 bits).

Etapa de Segmentación del Pentium:





Tema 2. Memoria Virtual

1. Introducción a la memoria virtual (MV)

- Es un esquema de gestión de memoria en el que los procesos:
  - Se ejecutan sin estar completamente cargados en memoria principal (MP).
  - Pueden tener mayor tamaño que la MP disponible.
- Permite un desacoplamiento entre el espacio de direcciones físicas y el espacio de direcciones virtual.
- **Utiliza el almacenamiento secundario como extensión de la MP**.
- Basada en el principio de localidad de las referencias → **Sólo se mantiene en MP la información necesaria en cada momento**.
- El control lo realiza el sistema operativo (SO) con la ayuda imprescindible del hardware.

Ventajas de la MV:

- Transparencia en las transferencias, dispositivo de almacenamiento ↔ MP.
- **El tamaño de los procesos depende del espacio de direccionamiento virtual y del disco duro.**
- Se mejora el rendimiento del sistema → se incrementa el **grado de multiprogramación** (nº de procesos cargadas en memoria principal, es decir, en ejecución).
- Reduce la E/S → sólo se cargan en MP las partes necesarias de un programa.

2. Conceptos relacionados con la MV

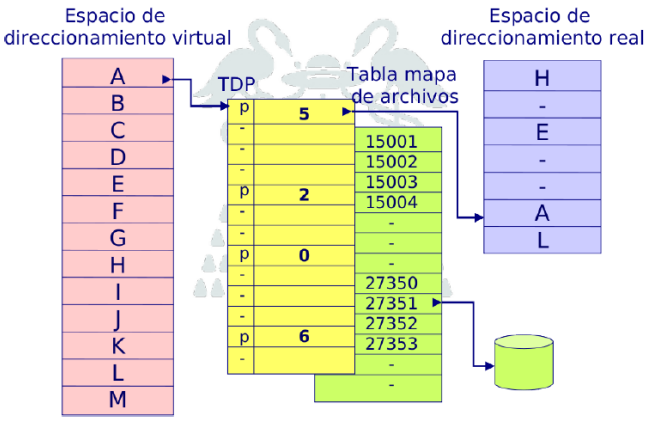
Requisitos hardware de la MV (los de la MMU)**:**

- Se utilizan mecanismos de **paginación** y **segmentación**.
- Ventaja de la paginación:
  - Transferencias más simples con bloques de tamaño fijo. Disco ↔ MP.
- Hardware de paginación:
  - Unidad de gestión de la tabla del mapa de páginas.
  - Bits en los descriptores de páginas de: presencia, modificación (dirty bit) y referencia.
  - Almacenamiento auxiliar para las páginas del proceso.
  - Soporte para interrumpir instrucciones.

Notas:

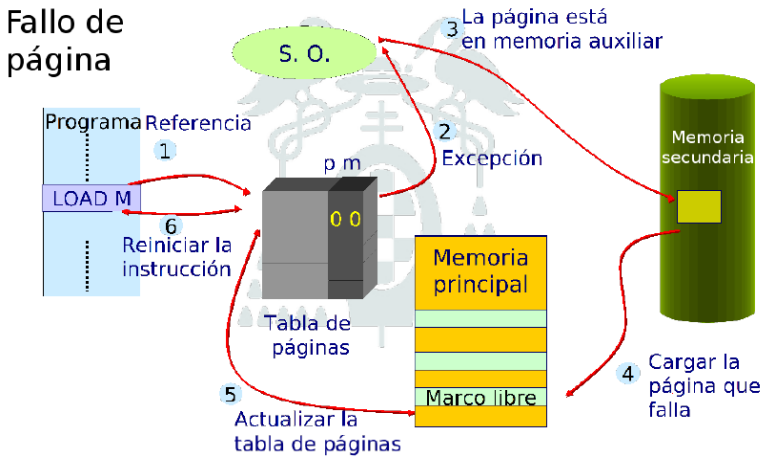
- El **CP** apunta a una instrucción.
- El acceso a disco es muy lento, durante ese periodo se está produciendo un cambio de contexto (otro proceso se ejecutará).
- El soporte de interrumpir instrucciones vuelve a colocar el CP en la posición que se quiere ejecutar cada vez que se produce un **fallo de página**.
- Se produce un fallo de página cuando no está en **MP**, después del fallo de página se ejecuta otra instrucción.
- El CP se incrementa en cuanto se procesa el código de instrucción (antes de saber si tiene los operandos o no.

¡La TDP se almacena en MP!



Carga dinámica:

- Transferencia de páginas de dispositivo de almacenamiento a MP.
- Si la página referenciada no está disponible en MP se produce un **Fallo de Página** (FP). Cuando se produce un FP tenemos que activar el DMA y cargar la página que está produciendo el fallo, de disco duro a MP → **carga dinámica**.
    - La tasa de fallos de página disminuye al aumentar el número de marcos.
- Los tiempos que más afectan a la carga dinámica son:
    - Cambios de contexto (=cambio de proceso). Consiste en restaurar el estado de programa…
    - Guardar una página modificada en disco (**page out**). Es lo que más afecta.
    - Cargar una página referenciada en MP (**page in**).
- Mientras se realiza, el proceso está bloqueado.



Paginadores (procesos del SO que realiza la carga dinámica → tratamiento de la excepción de FP):

- Parte del SO que mueve páginas entre disco y MP.
- Rutinas para hacer la transferencia cuando se produce un FP.
- Se crean y destruyen con el objeto proyectado en MV.

Hay varios tipos de paginadores:

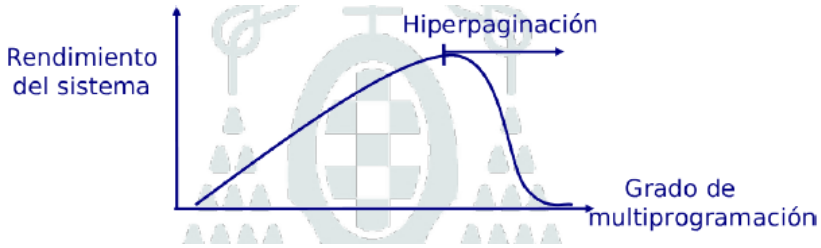
- **De archivos**:
    - Por ejemplo: mmap, exec (carga el programa en MV).
- **De objetos anónimos o swap pager**. Aquellos que no tienen una imagen en el sistema de archivos.
    - Gestión del área de swap (área de memoria persistente para los objetos anónimos).
- **De dispositivos**:
    - Por ejemplo: gestión del *frame buffer*.
    - Se proyecta la zona de memoria utilizada por el dispositivo (no en el disco).

¡Los registros se guardan en el BCP (bloque de control de procesos)!

¡El PC es actualizado por el dispatcher!

Hiperpaginación, vapuleo o *thrashing*

La hiperpaginación es la caída de rendimiento al aumentar el grado de multiprogramación. Cuando los procesos intercambian páginas disco ↔ MP.

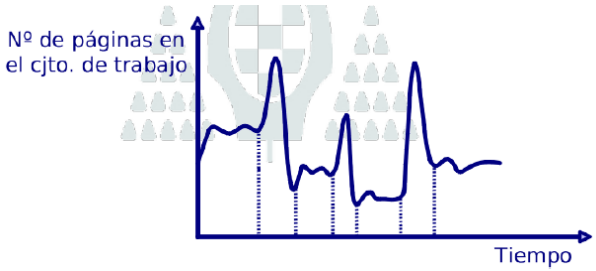


Las soluciones son:

- **Reducir el grado de multiprogramación** (cerrar programas).
- **Utilizar un algoritmo de reemplazo local** (algoritmos de reemplazo con política LRU).
- **Suministrar a cada proceso el nº de marcos que necesite**.
    - Modelo del conjunto de trabajo.
    - Frecuencia de fallos de página.

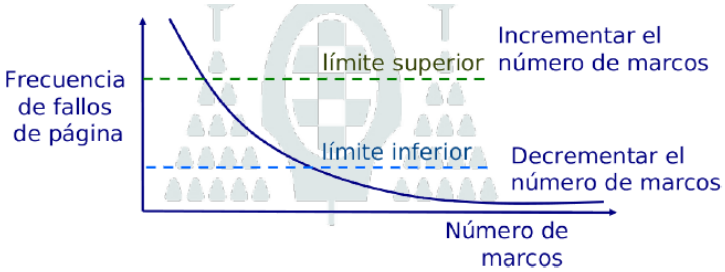
Modelo del conjunto de trabajo:

Conjunto de páginas referenciadas por el proceso durante un intervalo de tiempo. Está basado en el principio de localidad de referencias y mantiene alto el grado de multiprogramación.



Frecuencia de fallos de página:

Se establece un umbral superior y otro inferior, para quitar o asignar marcos de página a un proceso.



3. Algoritmos de gestión de la MV

El objetivo de los **Algoritmos de gestión de memoria** es minimizar el porcentaje de FPs, con mínima sobrecarga y máximo aprovechamiento de la MP.

Las **Políticas de asignación** determinan que cantidad de MP se asigna a un proceso según sus necesidades.

- **Asignación fija**: El número de marcos se decide en la carga inicial y esta determinado por el tipo de proceso.
- **Asignación variable**: El número de marcos cambia a lo largo de la vida de un proceso (modelo del conjunto de trabajo y frecuencia de FPs).
- **Alcance del reemplazo Global**: Considera todos los marcos de MP como candidatos, independientemente del proceso.
    - Ventaja: Mejor aprovechamiento de la MP; Inconveniente: Hiperpaginación; Linux.

- **Alcance del reemplazo Local**: Considera los marcos del proceso que origino el FP.
    - Ventaja: El numero de FPs es mas determinista; Inconveniente: Mayor sobrecarga por el calculo de los marcos a asignar en cada instante; VMS y Windows.
- **Gestión del espacio libre**: El SO mantiene el estado de los marcos libres o asignados. (Mapa de bits), Listas enlazadas (Windows), Sistema de colegas o buddy (Linux, Unix)).

Las **Políticas de ubicación** determinan donde se ubica un bloque en MP. Se utiliza sobre todo en segmentación, ya que, en paginación el tamaño de los bloques es el mismo. Se utiliza el primer ajuste, siguiente ajuste, mejor ajuste y peor ajuste (es el que menos fragmentación externa provoca, pero conlleva mucho tiempo).

Las **Políticas de búsqueda o de lectura** determinan cuando y que paginas se cargan en MP.

- **Paginación por demanda**: Solo se carga en MP cuando se ha referenciado, así en MP solo hay lo que se necesita y la sobrecarga es mínima.
- **Paginación anticipada o prepaginacion**: Se cargan en MP según una predicción. Si la predicción es buena, el tiempo de ejecución de los procesos se reduce y es útil cuando se accede secuencialmente a dispositivos de almacenamiento.

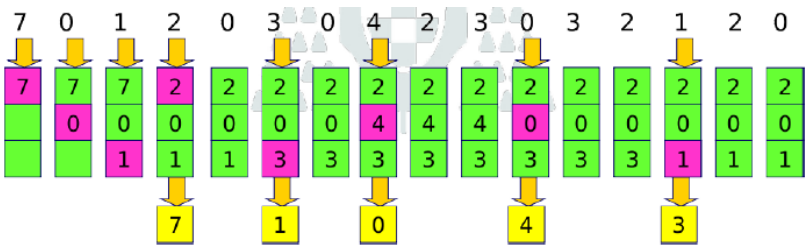
Las **Políticas de reemplazo** deciden que paginas deben sustituirse en MP. Los criterios a seguir son la baja sobrecarga, no **tuning** (ajustes en máquinas con distintas configuraciones) y aproximación al LRU.

La **cadena de referencias** (Ej.: el primer acceso a la página 1 será fallo) es una lista de referencias a paginas para evaluar la calidad de los algoritmos, eliminando los accesos contiguos a la misma pagina y se obtienen artificialmente, de forma pseudoaleatoria ó a partir de una traza de ejecución.

Cadena de referencias = páginas que se van referenciando (quitar desplazamiento y aquellas que se repiten no se referencia).

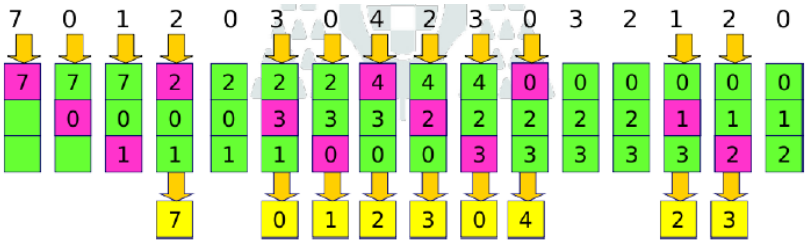
Algoritmo optimo

Se reemplaza la página que va a tardar mas tiempo en usarse. Ofrece la tasa de fallos mas baja posible pero no se puede implementar



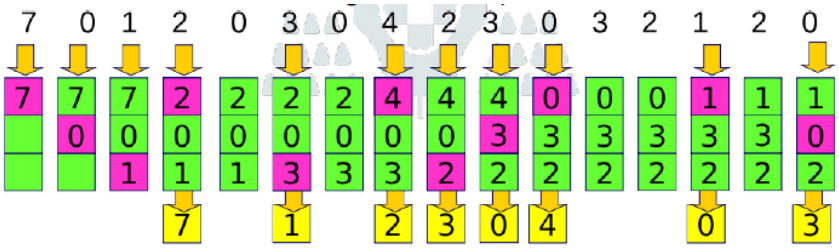
Algoritmo FIFO

Se reemplaza la página que lleva más tiempo cargada en MP. Es sencillo de implementar, pero tiene un bajo rendimiento y se puede dar la **Anomalía de Belady** (incremento de FPs al aumentar el nº de marcos).



Algoritmo LRU

Se aproxima al algoritmo óptimo. Utiliza el pasado reciente para predecir el futuro. Sustituye la pagina menos usada en el pasado. Su implementación es compleja y **tiene una alta sobrecarga**. La solución a estos inconvenientes es la utilización de algoritmos de aproximación al LRU.



Cada vez que se accede a una página se actualiza el **timestamp** tanto si es FP o si es acierto.

Aproximación a LRU

Reloj Global, FIFO con segunda oportunidad y NFU. Para todos los casos eliminamos el **timestamp** y trabajamos con un **bit de referencia**.

Reloj Global

Se crea una lista circular y se emplea un bit R asociado a cada página.

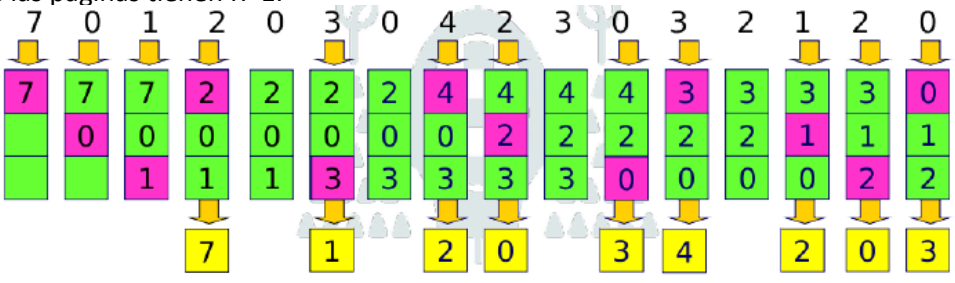
- Si la lista no está llena carga páginas con R=0.
- Si se referencia una página R=1.
- Cada cierto periodo, con un puntero giratorio pone R=0.
- Si la lista está llena. Si R=0 se reemplaza la pagina y avanza el puntero. Si R=1, pone R=0 y avanza el puntero.

FIFO con 2ª oportunidad

Se emplea un bit R asociado a cada página. (Si no hay FP no aplicamos el algoritmo, **no modificamos el puntero, pero modificamos el bit de referencia**)

- Se elige una página con criterio FIFO.
- Si R=1 poner R=0.
- Si R=0 sustituir la página.
- Avanzar el puntero e ir a 1.

Generalmente se implementa con una cola FIFO circular. Tiene baja sobrecarga, pero puede degenerar en un FIFO si todas las paginas tienen R=1.



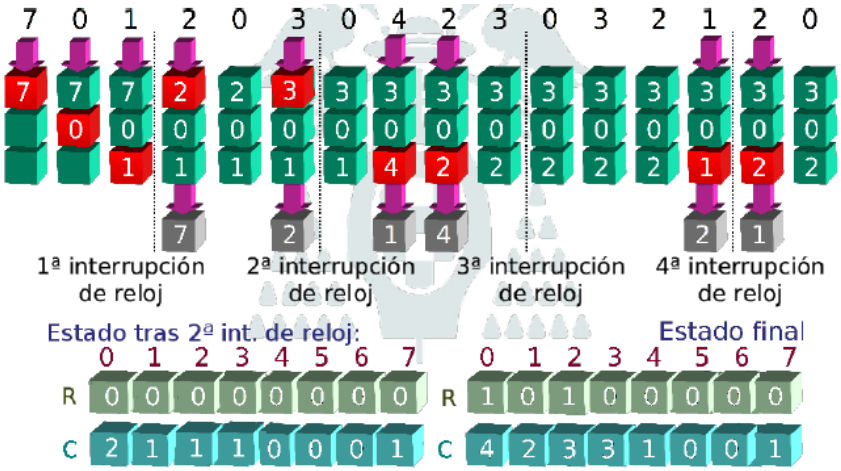
NFU

Controla la interrupción de un reloj y se emplea un contador y un bit R asociado a cada página.

- Por cada interrupción del reloj. Si R=1, incrementa el contador y pone todas las R=0.
- Por cada FP, se reemplaza la pagina con el menor valor en el contador.

El inconveniente es que si una pagina se usó mucho, no se reemplazará, aunque ya no se acceda a ella. La solución es emplear **mecanismos que envejezcan los contadores**. En vez de incrementarlos:

- Desplazar el contador 1 bit a la derecha.
- Añadir bit de referencia en el extremo izquierdo del contador.



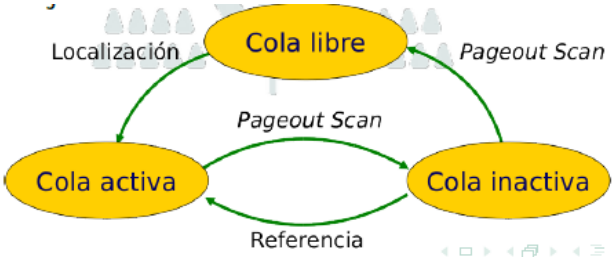
Casos de estudio

Mach 3.0

Utiliza la aproximación al LRU con FIFO segunda oportunidad (no la misma que la estudiada anteriormente). Utiliza tres colas de páginas:

- **Cola de paginas libes**, puestas a 0. Se reemplazan cada vez que hay un FP. (Son las que no están cargadas en MP y no están siendo utilizadas muy a menudo. P=0 y R=0).
- **Cola de paginas activas**. Mantienen el conjunto de trabajo. (Son las que forman parte del conjunto de trabajo. P=1 y R=1. Consumen MP).
- **Cola de paginas inactivas.** Buffer de paginas no referenciadas. (Son las que están cargadas en MP, pero llevan mucho sin estar referenciadas. P=1 y R=0).

El demonio (programa que se ejecuta cada cierto tiempo) pageout libera páginas, moviéndolas de la cola de activas a la de inactivas cuando el numero de paginas libres cae por debajo de un umbral.



Windows (W2K)

Mecanismo de clustering:

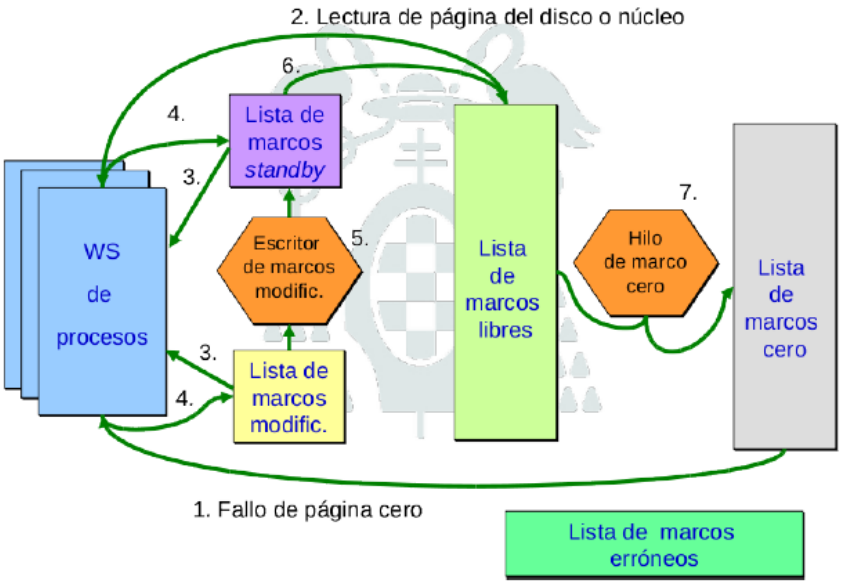
- Si hay FP, realiza paginación anticipada o prepaginacion, trayendo varias páginas contiguas
- Depende de la cantidad de MP y tipo de objeto que produjo el FP
- En sistemas de sobremesa el valor del clustering es 8 marcos para código, 4 para datos y 8 para el resto. (TODO ESTO SUCEDE CADA VEZ QUE HAY UN FP).

Gestión del conjunto de trabajo (WS) → Numero de paginas cargadas en MP del proceso en ejecución:

- Asignación variable de numero de marcos y alcance local.
- Al iniciar un proceso se asignan un tamaño mínimo del WS. Que varía dinámicamente entre 50 y 345 paginas.
    - Reajusta el WS de los procesos y decide que cantidad de paginas se pueden liberar. E incrementa el tamaño del WS si el proceso genera FPs.

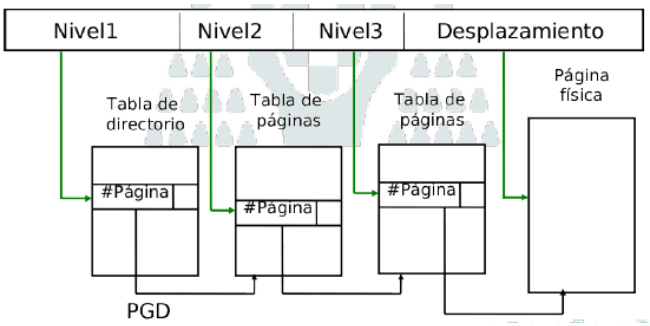
Los marcos pueden tener varios estados:

- Erróneo → Con daños físicos.
- Libre → Sin asignar. Pero no a 0s.
- Vacío → Iniciado a 0s.
- Activo → Presente en un WS.
- Transición → Está actualizándose con información de disco.
- Reposo → Ha dejado de pertenecer a un WS, pero contiene información y esta apuntado desde la TMP.
- Modificado → La información que contiene no se ha salvado en disco.
- Modificado sin escritura → Es necesario esperar para actualizar su información en disco.



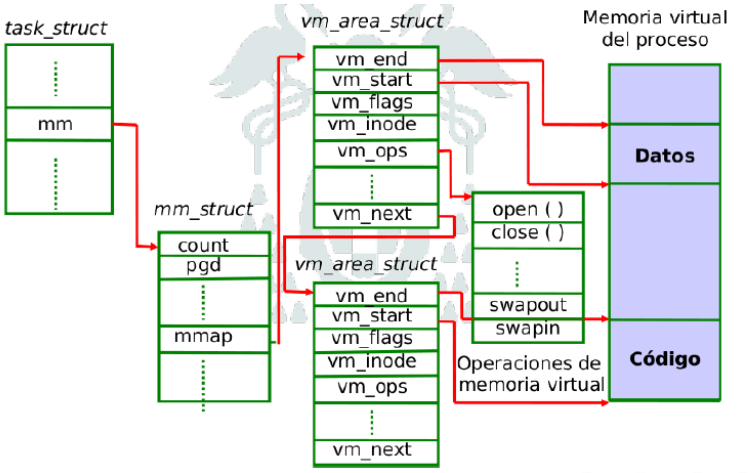
Linux

Direccionamiento de la MV

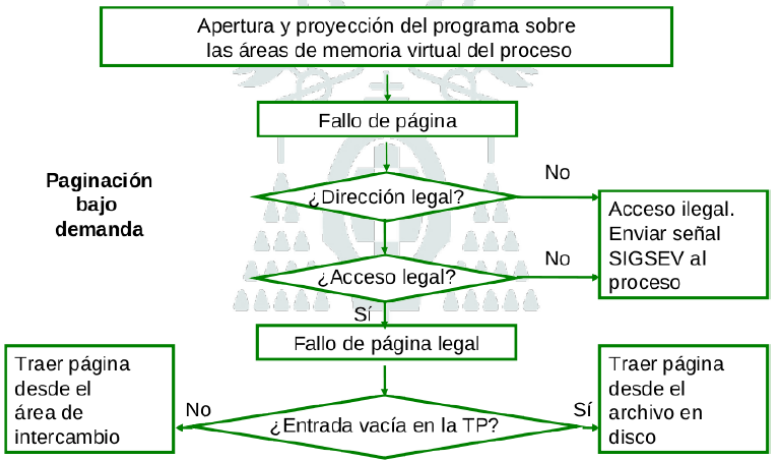


- Tabla de páginas con tres niveles y cuatro campos: **directorio de páginas**, **directorio intermedio**, **tabla de páginas** y **desplazamiento**.
- Cada tabla de páginas ocupa 1 página.
- Diseñada para Alpha 64 bits. Se adapta a x85 (32 bits) definiendo tamaño del directorio intermedio=1

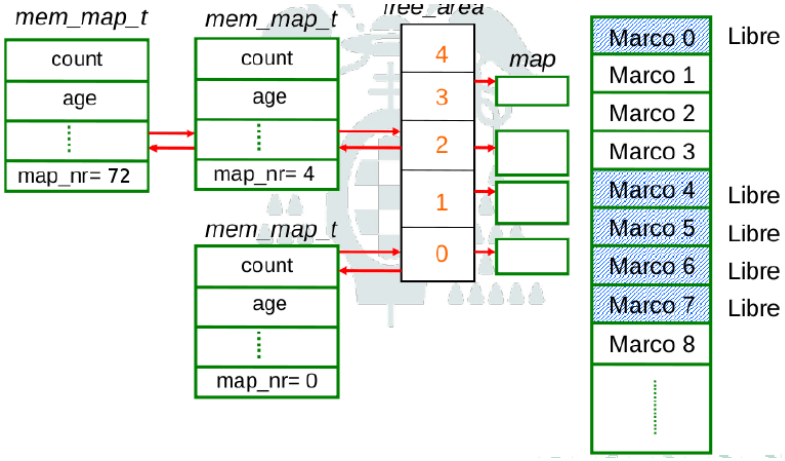
Estructura de datos:



Fallo de páginas:

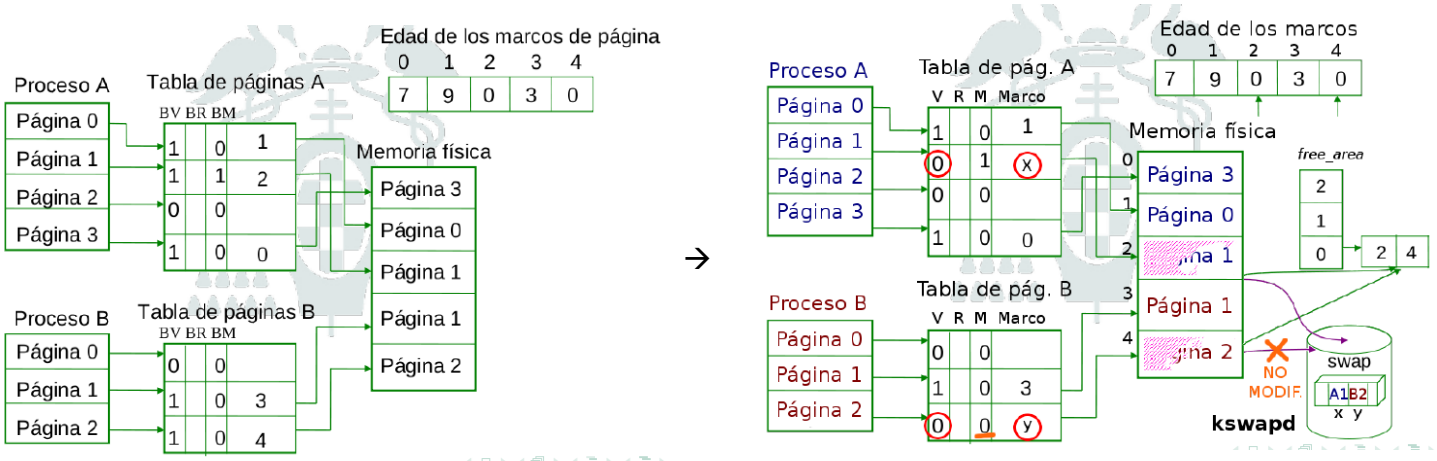


Asignación de páginas utilizando el sistema de colegas:



Algoritmo de reemplazo:

- Alcance de reemplazo global.
- La gestión del área de intercambio se realiza con el demonio ***kswapd.***
    - Cada segundo comprueba el nº de marcos libres y busca los que pueden ser reemplazados.
    - Es un algoritmo de aproximación al LRU con envejecimiento.
    - Basado en la edad de las páginas.
- Técnica de envejecimiento de páginas.
    - Todas las páginas se inician con edad 3.
    - Si se accede a una página, R=1, se incrementa en 3 la edad de la página, hasta un máximo de 20.
    - Si se ejecuta ***kswapd***, se decrementa en 1 la edad de las páginas que no se usan (bit R=0).
- Si una página modificada se lleva a disco:
    - Se marca como inválida la entrada en la TMP.
    - Se incluye la información para su recuperación posterior.
    - Se libera el marco, añadiéndolo a lista de marcos libres.
- Las páginas no modificadas se marcan como libres y sus marcos se añaden a la lista de marcos libres.
- Si se recuperan suficientes marcos para el proceso, el demonio dormirá de nuevo.
- Si no, se continúa con el siguiente proceso.



Pequeños Apuntes

Cuando se produce la excepción de fallo de página se lleva a cabo la **carga dinámica**. El hardware necesario es un **circuito**, **TDP** (MP) y **Tabla de Mapa de Archivos** (MP).

El **BCP** nos proporciona información como por ejemplo donde empieza la TDP.

Cuando se produce un FP (bit de presencia=0), se carga el proceso en la tabla de mapa de archivos, y se busca el proceso en el disco duro para cargarlo en MP y actualizar la TDP (bit de presencia=1).

Un **demonio** es un proceso del sistema que se utiliza periódicamente.

En Linux el BCP es **Task Struct** y el **memory map** apuntará (?) al área de datos en MV. **free-area** es un array.

TEST DE EXAMEN

1. Los sistemas con segmentación paginada evitan la existencia de:
   - **a.** Fragmentación externa.
   - **b.** Ni fragmentación interna ni externa,
   - **c.** Fragmentación interna.
   - **d.** Tanto fragmentación interna como externa.
2. ¿Qué campos de un bit es habitual almacenar en las entradas de las tablas de páginas?
   - **a.** Todos los mencionados en las demás respuestas (PMR y RWX).
   - **b.** Solo estos: lectura(R), escritura (W), ejecución (X).
   - **c.** Solo estos: presente (P) y modificado (M).
   - **d.** Solo este: referenciado (R).
3. Sea un sistema de gestión de memoria con segmentación paginada. Se sabe que el tamaño máximo del espacio de direccionamiento virtual de un proceso es de 4GB y que el tamaño de marco y de página es de 4KB. Indique cuál podría ser el formato de una dirección virtual
   - **a.** Segmento: 8 bits, página: 12 bits, desplazamiento: 12 bits.
   - **b.** Segmento: 4 bits, página: 16 bits, desplazamiento: 12 bits.
   - **c.** Segmento: 12 bits, página: 8 bits, desplazamiento: 12 bits.
   - **d.** Todas las respuestas son correctas.
4. Si una determinada arquitectura utiliza 16 bits para la dirección física y marcos de página de 512KB ¿Cuál afirmación es cierta?
   - **a.** Habrá como máximo 32 marcos de página.
   - **b.** Habrá como máximo 64 marcos de página.
   - **c.** Habrá como máximo 16 marcos de página.
   - **d.** No es posible utilizar marcos tan grandes con solo 16 bits en el bus de direcciones.
5. La unidad de gestión de memoria de una arquitectura Pentium permite:
   - **a.** Establecer el límite de los segmentos 2^{20} páginas.
   - **b.** Establecer el límite de los segmentos 2^{20} bytes.
   - **c.** Todas las respuestas son correctas.
   - **d.** Que los segmentos puedan reubicarse y crecer dinámicamente.
6. En un sistema con MV basada en segmentación pura, indique cuál de las siguientes afirmaciones es correcta con respecta a 2 procesos que comparten un único segmento de su espacio de direccionamiento:
   - **a.** 2 direcciones virtuales **distintas**, una de cada proceso, pueden referenciar la misma dirección física.
   - **b.** Cada proceso tiene un espacio de direccionamiento virtual independiente, incluso para el segmento compartido.
   - **c.** Todas las respuestas son correctas.
   - **d.** Un bloque de MP puede ser accedido por los 2 procesos.
7. Complete la siguiente afirmación correcta. “La fragmentación externa …”
   - **a.** Sólo puede solucionarse en caso de que el sistema soporte reubicación estática.
   - **b.** Puede solucionarse utilizando técnicas de compresión de memoria.
   - **c.** No puede solucionarse.
   - **d.** Puede solucionarse utilizando técnicas de compactación de memoria.

8. ¿Es necesario que haya un campo límite en las entradas de las tablas de páginas?
    a. No, porque todas las paginas tienen el mismo tamaño.
    b. Todas incorrectas.
    c. No, porque en paginación las direcciones virtuales no tienen campo desplazamiento.
    d. Si, igual que en las tablas de segmentos.
9. ¿Por cuál de los siguientes motivos una entrada en la tabla de páginas podría tener el bit P (presencia) desactivado?
    a. Porque la página nunca llegó a ser cargada.
    b. Todas las respuestas son correctas.
    c. Porque no está cargada en un marco de MP.
    d. Porque dicha página fue desalojada por el sistema de memoria virtual.
10. Indique la respuesta correcta con respecto al principio de localidad de referencias a memoria:
    a. La ejecución del código de un programa puede apoyar la existencia tanto de localidad espacial como temporal.
    b. Todas las respuestas son correctas.
    c. El acceso a los datos de un programa en ejecución justifican la existencia de localidad espacial.
    d. Los procesos tienen a referenciar posiciones de memoria cercanas durante un breve intervalo de tiempo.
11. La existencia y utilización de pilas en los procesos justifica la presencia de localidad…
    a. Tanto temporal como espacial.
    b. Espacial.
    c. Temporal.
    d. Ni temporal ni espacial.
12. Indique cual de las siguientes afirmaciones es correcta:
    a. Todas las respuestas son correctas.
    b. Cada proceso tiene un espacio de direccionamiento virtual independiente al resto de procesos del sistema.
    c. En un sistema operativo multiproceso, el microprocesador genera direcciones virtuales
    d. El espacio de direccionamiento físico es compartido por todos los procesos del sistema, tanto los de usuario, como lo del núcleo del sistema operativo.
13. Indique cuál de las siguientes afirmaciones es correcta respecto a las regiones del mapa de memoria de un proceso en un cierto sistema operativo y para una determinada arquitectura:
    a. Se crean simultáneamente con la generación del archivo ejecutable asociado al proceso.
    b. Su tamaño es fijo: no se puede modificar durante la existencia del proceso.
    c. Hacen referencia a información sólo del archivo ejecutable asociado.
    d. La dirección lógica de comienzo de la región de código es idéntica para todo proceso.