Jerarquía de memoria

El SO multiplexa los recursos entre procesos en el tiempo. Cada proceso cree que tiene la máquina para el solo con su propia memoria y CPU. **El objetivo de la jerarquía de memoria es ofrecer a cada proceso su espacio virtual de memoria maximizando el grado de multiprogramación**.

Para lograr esto, el SO utiliza la llamada **memoria virtual**, donde la CPU genera direcciones virtuales (no se corresponden con ningún sitio físico de la memoria del ordenador) que son traducidas por la **MMU**. Dado que el mapa de memoria es virtual e inmenso para cada proceso, es imposible almacenarlo por completo en la memoria física. Por ello, se divide en **páginas** que se reparten entre la memoria física y disco. Cuando es necesario una pagina de la memoria principal, se produce un **fallo de página**, obligando al SO a cargar dicha pagina desde disco hasta memoria (costoso)

Traducción de direcciones

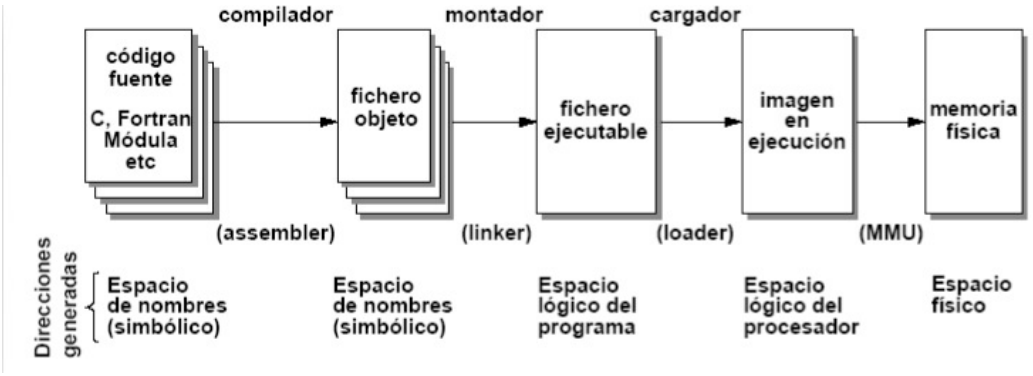
Existen dos tipos de direcciones con las que trabajan los SO actuales:

- **Espacio de direcciones lógico o virtual:** Son las direcciones que emite el procesador y que se almacenan en las direcciones y registros. El rango de estas direcciones depende de la cantidad de memoria asignada por el SO (Ej. un sistema de 64 bits tiene como máximo 2^{64} direcciones disponibles)
- **Espacio de direcciones físico**: El que aceptan los bancos de memoria del computador. Se corresponde con los bancos de memoria RAM, ROM y los controladores de E/S mapeados en memoria

Fases de traducción de direcciones

En los lenguajes de alto nivel el programador no usa direcciones numéricas, sino utiliza variables para identificar posiciones de memoria con datos y funciones para identificar posiciones de memoria con instrucciones. Estos nombres simbólicos se traducen a direcciones en distintas fases de traducción:

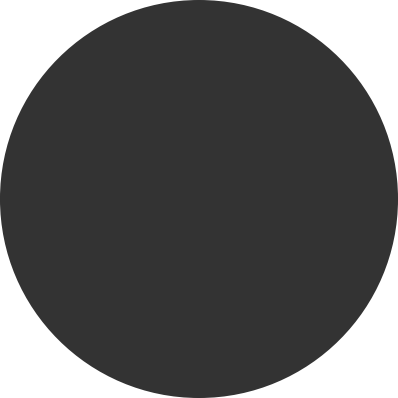
- **Compilación**: Separa los datos y el código. Genera código ensamblador con direcciones relativas al comienzo de las funciones para cada fichero de código objeto
- **Montaje**: Combina los ficheros de código objeto en un solo ejecutable. Resuelve referencias cruzadas (funciones de otro archivo). Crea las secciones de código y datos (Direcciones lógicas de programa)
- **Carga**: Asigna direcciones iniciales a las secciones del programa para la ejecución del código. Reubica en el espacio lógico (Direcciones lógicas de proceso)
- **Ejecución**: La MMU traduce direcciones lógicas a físicas en tiempo real (Direcciones físicas)

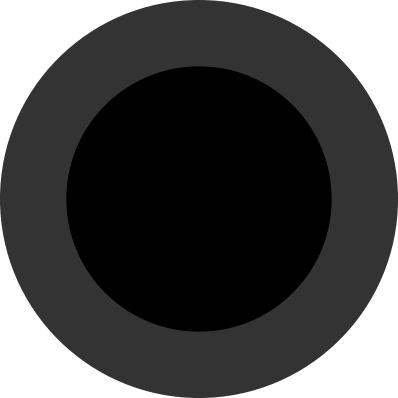


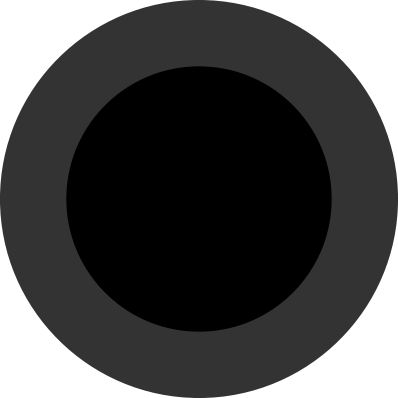
Modelos de gestión de memoria

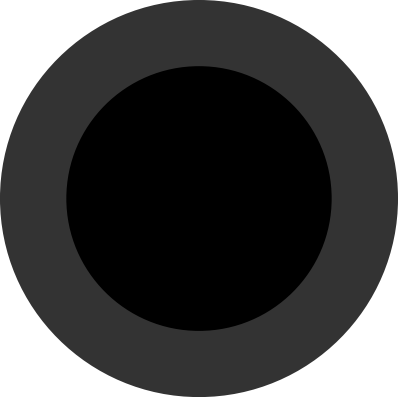
Según el número de procesos

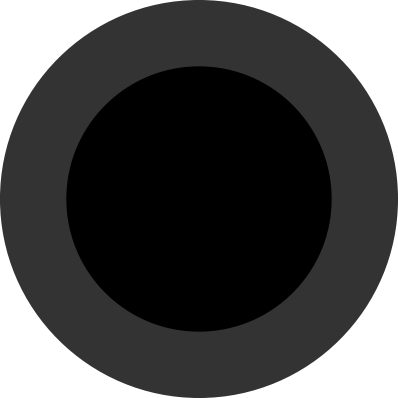
- **Uniprogramado**: Un único proceso
  - Sin reubicación: Enlazador genera direcciones físicas absolutas. Necesita saber el espacio que ocupará el SO: D.programa = D.proceso = D.Físicas
  - Con reubicación: El enlazador ubica el código entre 0 y k El cargador reubica el código en M (M-n = k)

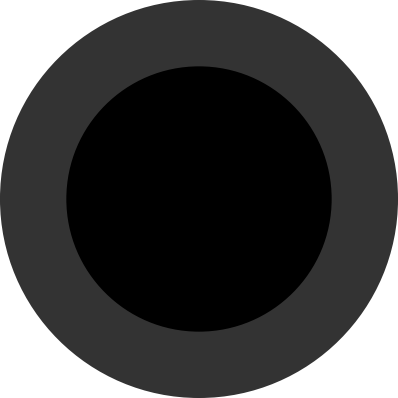


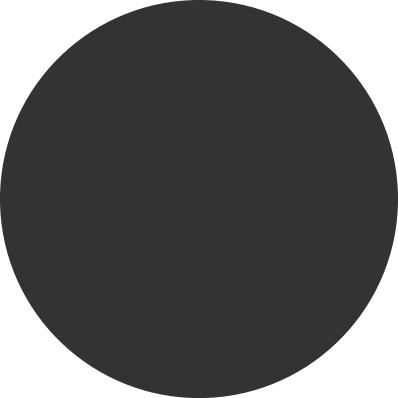


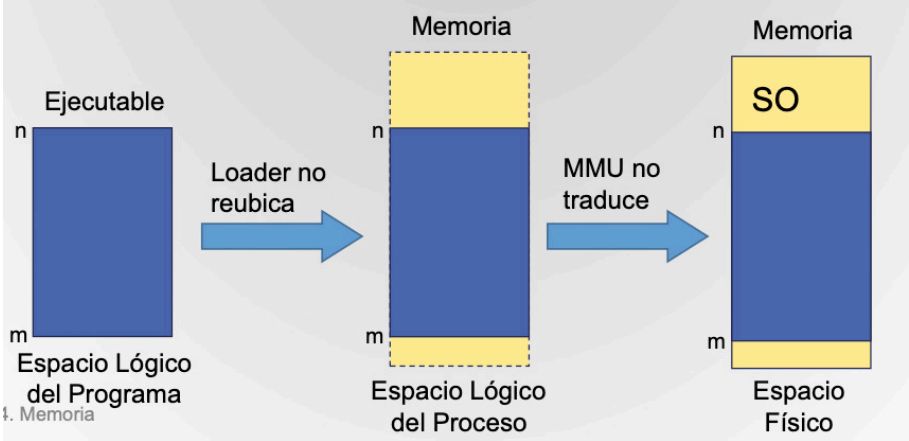


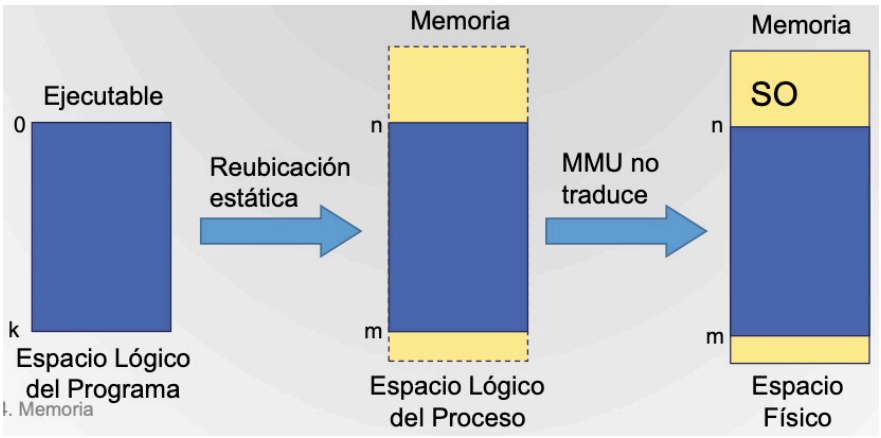












- **Multiprogramado**: En la PCB se almacena el límite superior e inferior al que el proceso puede acceder. Los procesos se crearán en zonas libres de memoria que el SO gestiona. Si un proceso trata de acceder a una dirección de memoria que no le pertenece se produce una excepción. Los procesos en modo kernel no lanzan estas excepciones. Estas zonas pueden ser de dos tipos
  - **Estáticas**: El SO divide la memoria física en particiones de tamaño fijo
  - **Dinámicas**: La memoria se va particionando poco a poco. Comienza como una única partición

Fragmentación

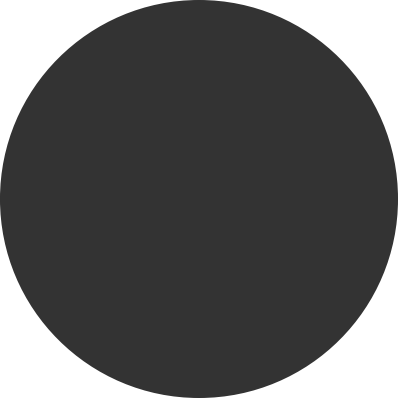
Se produce cuando no se aprovecha toda la memoria disponible. La fragmentación puede ser:

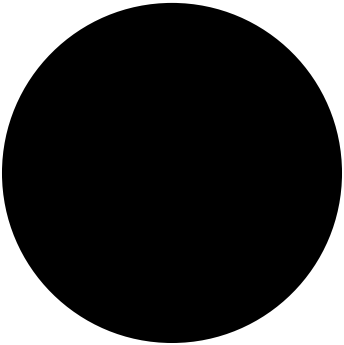
- **Interna**: Memoria asignada a un proceso que no se usa. Ocurre con bloques de memoria muy grandes
- **Externa**: Memoria libre entre bloques demasiado pequeña para ser usada

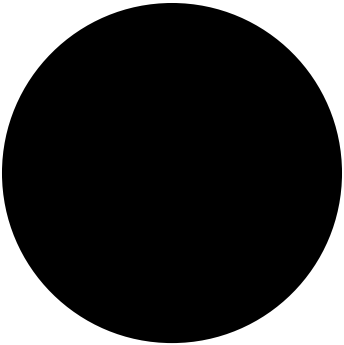
Políticas de asignación de bloques de memoria

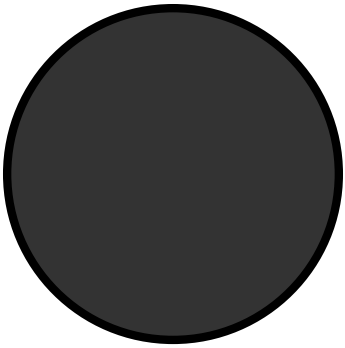
- **Particiones fijas**: Las particiones tienen siempre el mismo tamaño.
- **Particiones variables**: Particiones de distinto tamaño
  - **Best-Fit**: Elige aquella que mejor se ajuste
  - **Worst-Fit**: Elige aquella con mayor tamaño
  - **First-Fit**: La primera donde el proceso entre
  - **Next-Fit**: Como First-Fit, pero funciona como una cinta. Empieza a buscar desde el punto en el que lo dejó la ultima vez

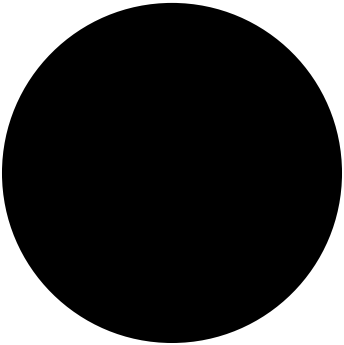
Según la ubicación de los procesos











Si el proceso se mantiene en memoria durante toda su ejecución o si se puede almacenar en disco temporalmente

- **Residente**: El proceso tiene que estar cargado en memoria desde que empieza hasta que termina. Si existe multiprogramación, el número de procesos está limitado por la capacidad de la memoria física
- **Modelo no residente**: Permite almacenar procesos en el disco y cargar de disco (swap-out y swap-in). Aumenta el grado de multiprogramación, pero los procesos de swap son muy lentos. Además, no todos los procesos pueden descargarse en memoria

Según el movimiento de los procesos

Si los procesos pueden cambiar de sitio (reubicación física)

- **Modelo inmóvil**: El proceso no puede cambiar la dirección de proceso una vez que el cargador reserva su memoria. Esto quiere decir que D.proceso = D.física
- **Modelo móvil**: Podemos cambiar un proceso de sitio (reubicarlo dentro de la memoria). Esto quiere decir que D.Proceso ≠ D.Física. Nos permite desfragmentar la memoria aprovechando más los recursos disponibles, por lo que aumenta el grado de multiprogramación

Según el orden de las direcciones físicas

Si los procesos ocupan posiciones consecutivas en el espacio físico

- **Modelo contiguo**: El proceso ocupa posiciones contiguas en el espacio de memoria. Es decir, todo el proceso forma un único bloque de direcciones reservado
- **Modelo no contiguo**: El proceso no tiene por qué ocupar posiciones consecutivas en el espacio físico. Puede estar repartido en bloques disjuntos por la memoria, de tal forma que mientras que el espacio de direcciones lógico es contiguo pero el físico no. Se puede hacer de dos formas:
    - **Segmentación**: El espacio lógico se divide en fragmentos de distinto tamaño llamados segmentos
    - **Paginación**: El espacio lógico se divide en fragmentos de igual tamaño llamados páginas

Según la carga de los procesos

Si el proceso es completamente cargado en memoria antes de iniciar su ejecución o si se carga poco a poco

- **Modelo entero**: La imagen del proceso se carga por completo antes de iniciar su ejecución
- **Modelo no entero**: La imagen del proceso no tiene porqué estar cargada en memoria completamente. Basta con tener las secciones en uso. Permite tener más procesos en memoria y más grandes, puesto que no ocupan la memoria sin necesidad. Este modelo de memoria permite las **DDL**

Dynamic Link Libraries

Son archivos con instrucciones ejecutables cargadas por demanda de un programa por el SO para su uso. Pueden ser:

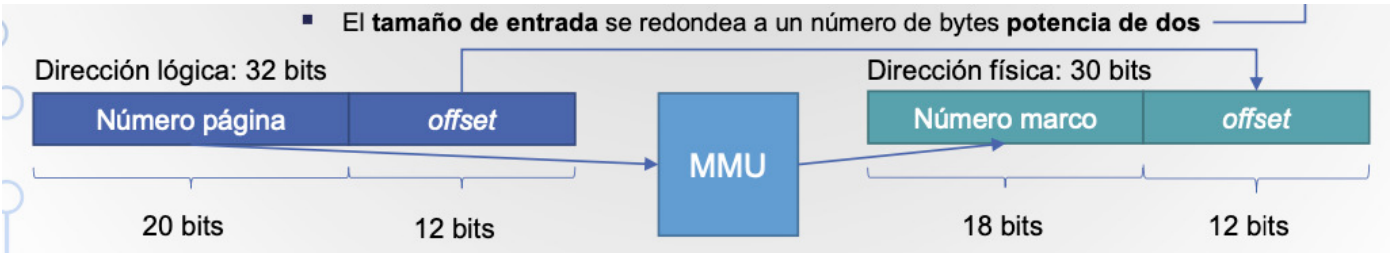
- **Con enlace estático**: El ejecutable incluye el código de todas las rutinas
- **Con enlace dinámico**: El ejecutable no incluye las rutinas. A cambio, se inserta un código que carga la rutina en memoria cuando es llamado por primera vez

Las librerías dinámicas permiten tener ejecutables que pesan menos y se elimina la redundancia. El código de las rutinas solo existe una vez entre todos los procesos. Estas rutinas serán necesarias cada vez que se ejecute dicho código

Paginación

- El espacio lógico es dividido en **páginas** de tamaño fijo. Cada página es una unidad de memoria compuesta por un número de posiciones consecutivas.
- El espacio físico es dividido en **marcos/frames** del mismo tamaño. Cada frame puede almacenar una página por completo

El SO mantendrá en memoria física una tabla de páginas para cada proceso con la tupla (página,marco). La MMU traduce la dirección virtual/lógica a física accediendo a la tabla de páginas del proceso. Se usa un registro especial que apunta al marco donde empieza la TP del proceso (**PTBR**) para ello. Cada acceso a dato o instrucción requiere dos accesos a memoria. Cada entrada de la TP contiene el marco correspondiente a la página y un bit de validez que indica si la página es del proceso o no. El *offset* dentro de la página es el mismo que dentro del marco



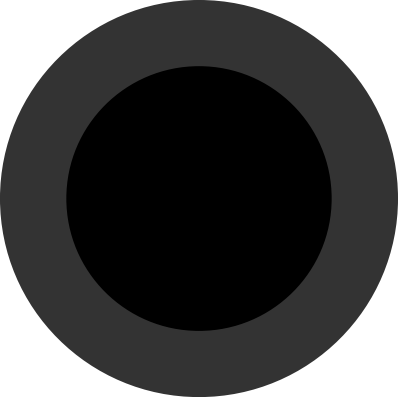
- **Ventajas**: Se pueden compartir páginas entre procesos. La TP de cada proceso puede tener punteros hacia las mismas páginas, indicando con determinados bits si son de solo lectura, lectura-escritura o privadas
- **Inconvenientes**:
  - La tabla puede ser enorme (TP invertida y paginación multinivel)
  - Tiempo de acceso mayor (TP + acceso a dato). Utilizamos una caché para la TP llamada **TLB**

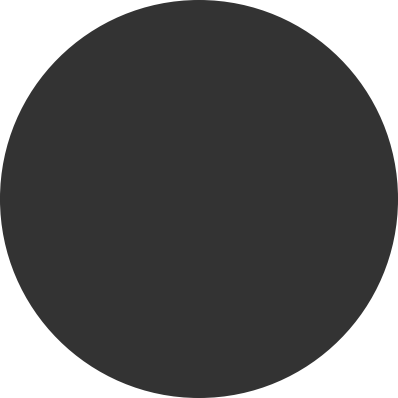
Tabla de páginas invertida

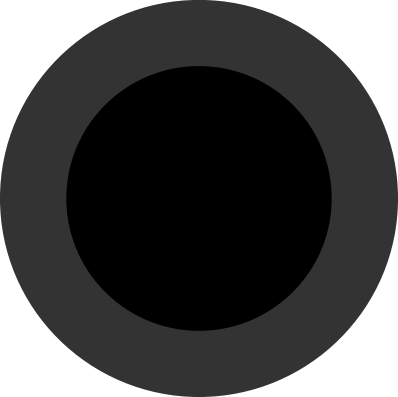
Las TP normales se almacenan en la PCB de cada proceso. Guardan la relación entre la página y su marco correspondiente. Las tablas de páginas invertidas mantienen una sola tabla de páginas del espacio físico para todo el sistema. Presenta los siguientes características:

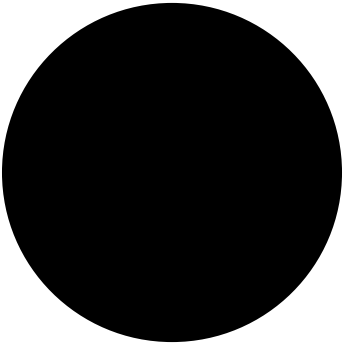
- Ocupa menos espacio
- Lento. Hay que recorrer toda la tabla para buscar la página que queremos
- Necesitamos el PID del proceso
- No almacena las páginas que no están en uso

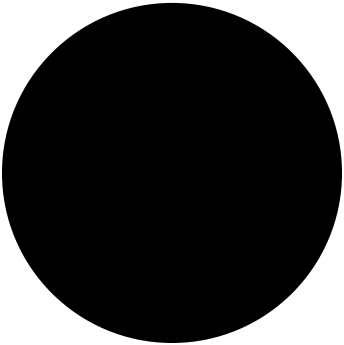
Marco 0	pidX	Nº página del proceso pidX en marco 0
…	…	…
Marco *n*	pidY	Nº página del proceso pidY en marco *n*

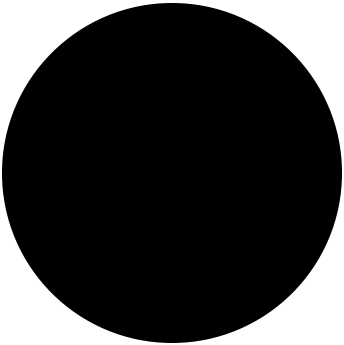


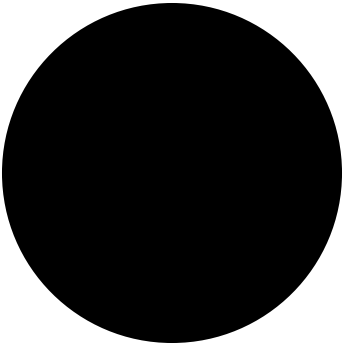












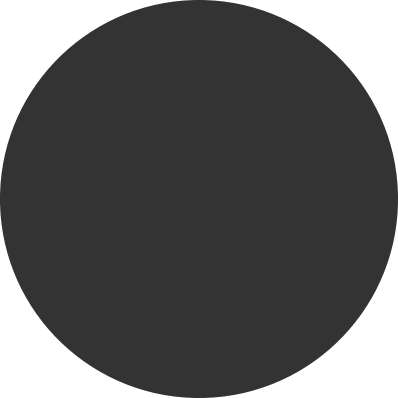
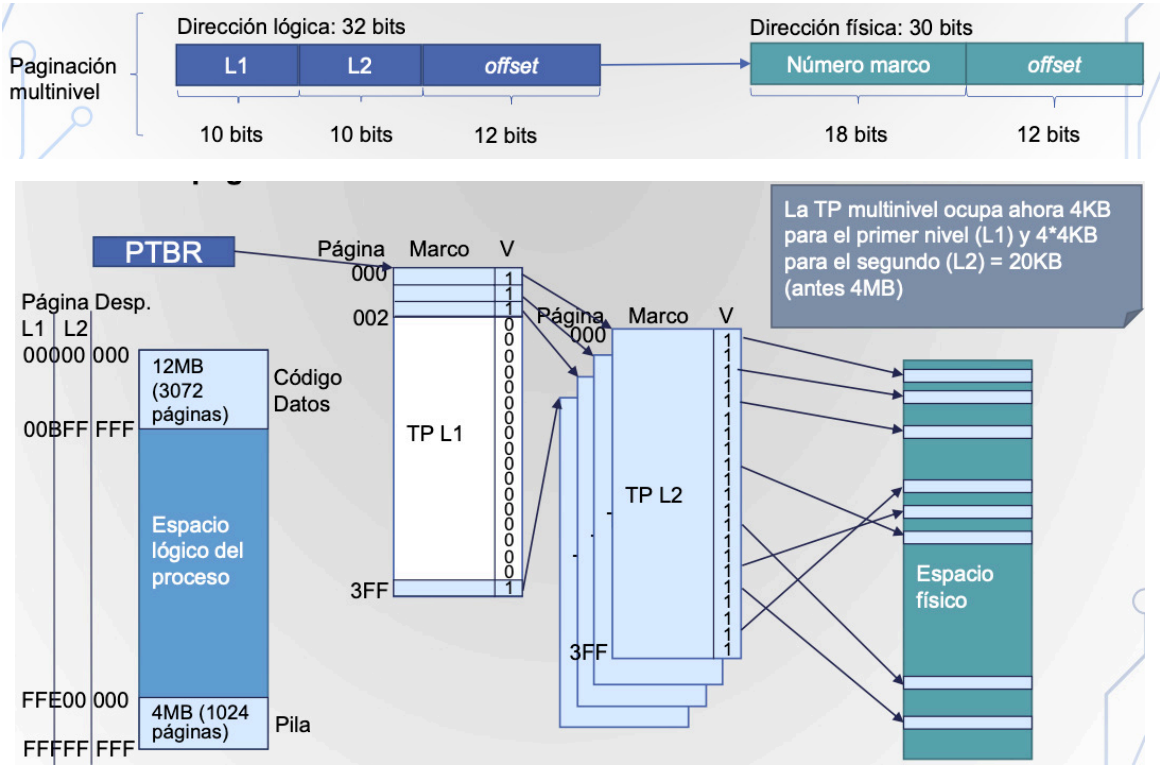


Tabla de páginas multinivel

Consiste en paginar la propia tabla de páginas para poder almacenarla en el disco usando dos niveles. Esto permite reducir el tamaño necesario de la tabla obligando solo a almacenar el primer nivel. El segundo nivel no tiene porqué mantenerse en memoria y puede almacenarse en disco. El funcionamiento es escalable y puede dividirse tantas veces como sea necesario



- **Ventajas**: Menor uso de memoria principal
- **Inconvenientes**: Acceso más lento (Acceso a 1º nivel, acceso a 2º nivel y acceso a marco)

Translation Lookside Buffer (TBL)

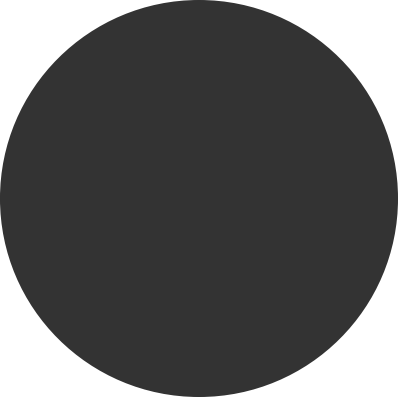
Una memoria caché encargada de almacenar la traducción de dirección lógica a física con aquellos más usados. Cada vez que se produce un cambio de contexto o bien se borra o bien se almacena el PID del proceso al que pertenece

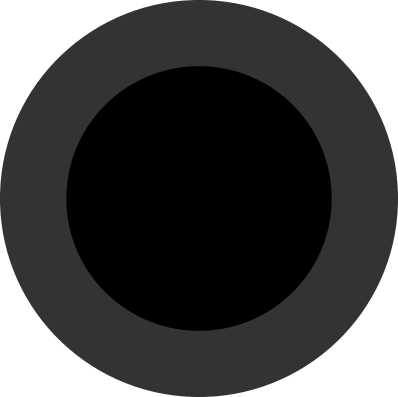
Segmentación

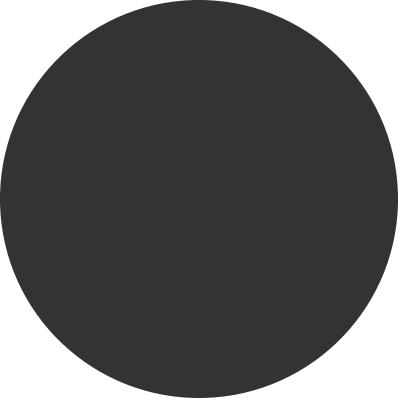
El espacio lógico se divide en segmentos de tamaño variable. De cada segmento se almacena su dirección base y su limite de tamaño (desplazamiento máximo). La MMU usa una tabla de segmentos (TS) con información sobre los segmentos, así como bits de protección (read, read-write,...).

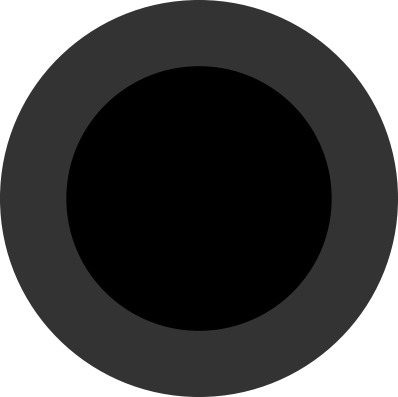
Segmentación paginada

- El espacio lógico del proceso se divide en segmentos
- Cada segmento se divide en páginas









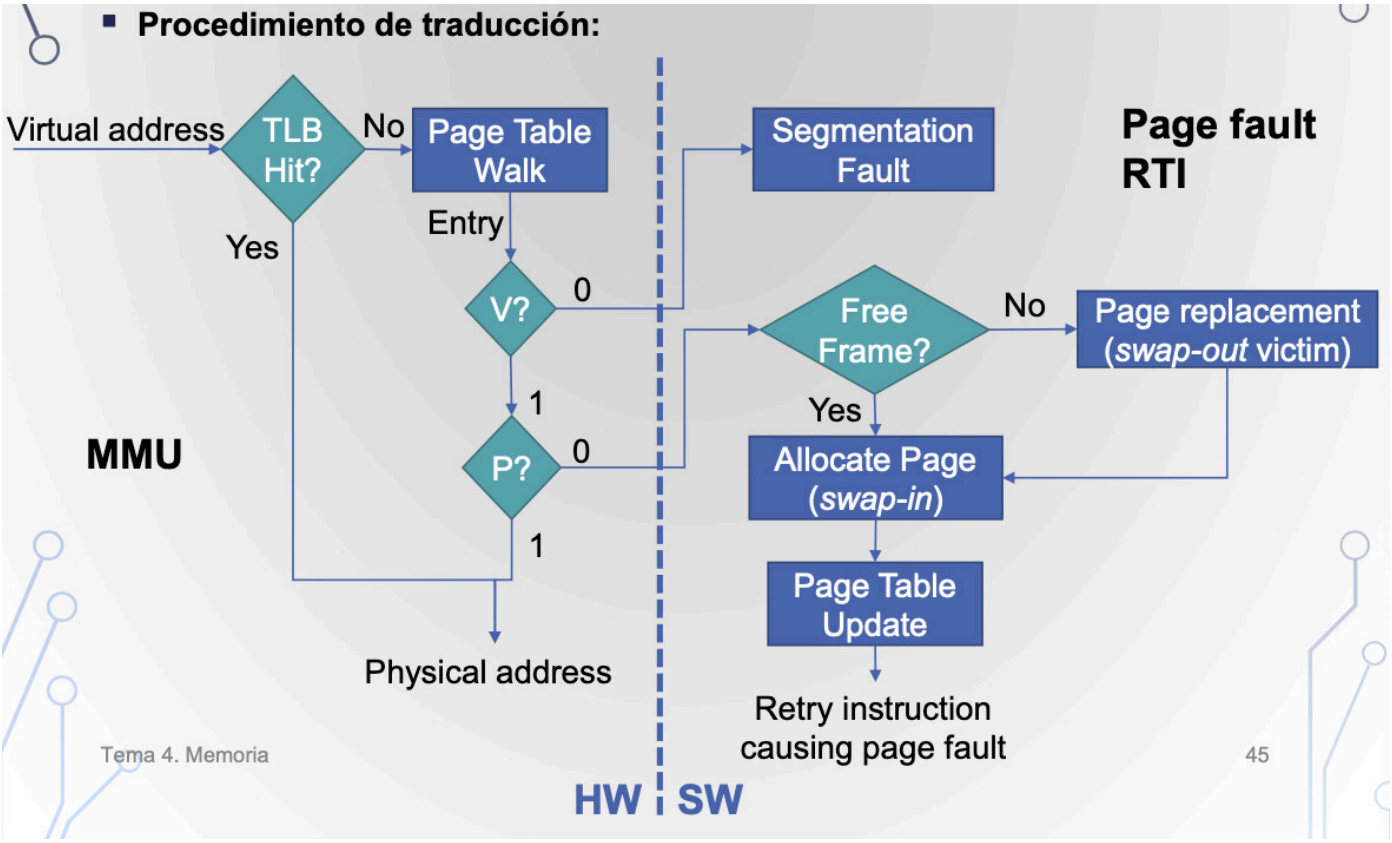
	Segmentación	Paginación
Tamaño de la dirección base	Dirección física completa al que hay que sumar el desplazamiento	Solo el número de página
Solapamiento (compartir memoria)	Se pueden solapar parcialmente segmentos físicos	Se pueden solapar páginas completas
Asignación de memoria	Complicado. N segmentos de tamaño adecuado. Es dificil hacer GC	Sencillo. Necesita N páginas de tamaño fijo

Memoria virtual

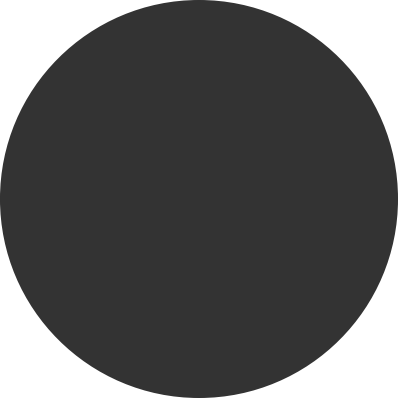
Se trata de una extensión de la jerarquía de memoria. Separa el espacio lógico del físico (el desarrollador ve un espacio extremadamente grande de memoria virtual). Es un modelo **no entero**, puesto que el proceso no necesita estar cargado completamente en memoria, solo aquella zona en uso. Esto permite aumentar el grado de multiprogramación y ofrecer la visión de una máquina con mucha memoria al programador

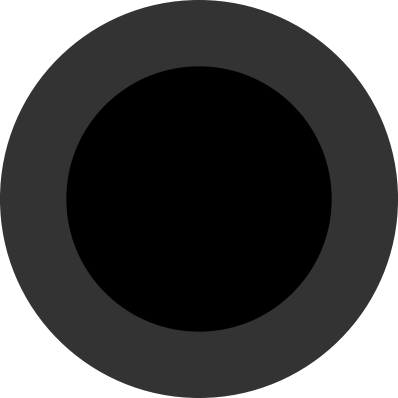
Requerimientos HW y SW

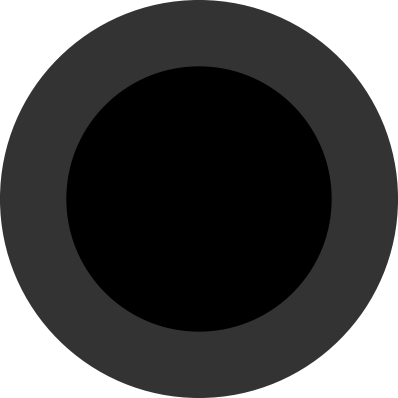
- **Unidad de transferencia de memoria virtual**
- **Bits de presencia (P), validez (V) y control (permisos, dirty,...) en la TP**
- **MMU**: Traducción de dirección lógica a física
- **Zona de swap**: Donde almacenar las páginas (partición del disco o ficheros del SO)

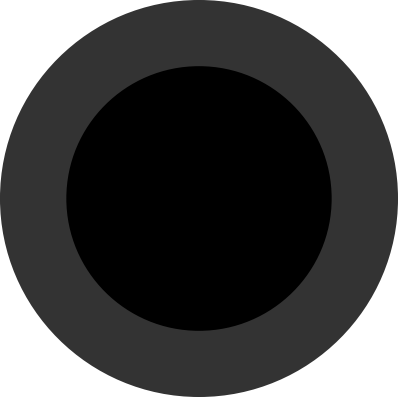


Algoritmos de asignación de marcos a procesos

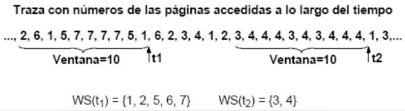
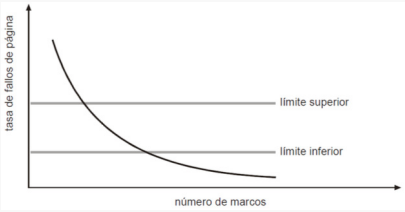
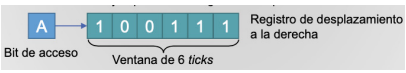
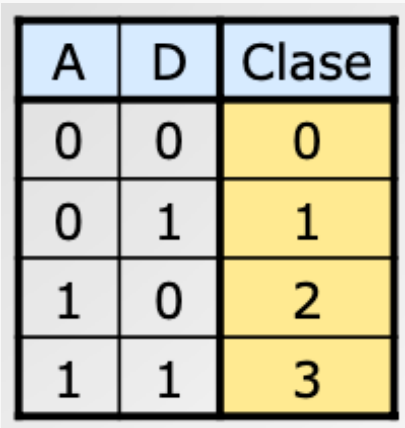


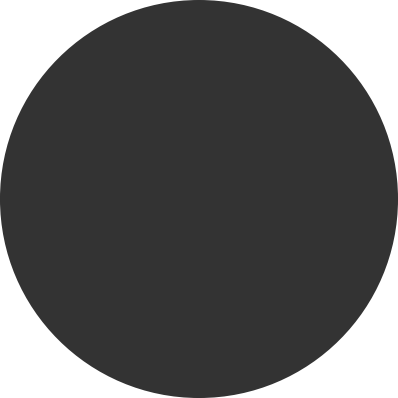


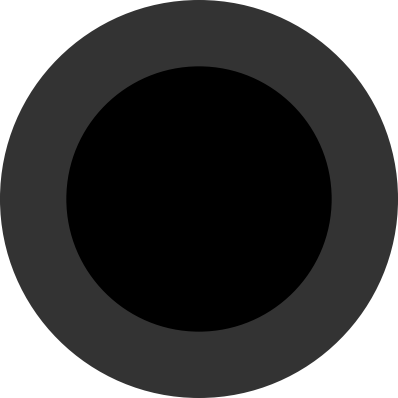




- **Fijo con reemplazo local**: Los marcos se reparten estáticamente entre procesos. A un proceso se le asignan N marcos. Cuando dicho proceso produzca un fallo de página, dicha página se cargará en uno de los marcos pertenecientes al proceso. De esta forma, su comportamiento es predecible y no afectará al resto de procesos
- **Dinámico**: Los marcos se asignan al proceso bajo demanda. Es una asignación que favorece a aquellos procesos con mayor número de fallos, ya que tendrán mayor número de marcos en uso
    - **Con reemplazo local**: Solo reemplaza páginas del propio proceso (working set)
    - **Con reemplazo global**: Todos los marcos son candidatos a ser reemplazados

Nombre	Descripción	Imagen
Working Set	Utiliza una ventana para calcular el número de marcos necesario para satisfacer las necesidades de localidad del proceso dentro de la ventana del tiempo	
PFF	Page Fault Frequency. Si la tasa de fallos de página es menor a un límite inferior se liberan marcos. Si es superior se asignan nuevos marcos al proceso	
Óptimo	Reemplaza aquella que no se va a usar más	
Aleatorio	Elige cualquiera	
FIFO	Primero en entrar, primero en salir	
LRU	Least Recently Used. Apunta hora de acceso en cada página o ordena la lista por orden de acceso (localidad temporal)	
Aproximación LRU	Bit de Acceso 8A) a cada una de las páginas. Cuando una página se referencia se pone el bit a 1. Cada n ticks de reloj se copia A en un registro de desplazamiento	
NRU	Not Recently Used. Utiliza los bits A y Dirty	
	Least Frequently Used. Dispone de un contador	





LFU	para cada marco. Dicho contador se incrementa en cada acceso	
NFU	Not Frequently Used. Acumula el bit A en un registro contador. Cada N ticks, si A está activo incrementa el registro contador. Se elige aquel con menor tamaño del contador	
FINUFO	First In Not Used, First Out. Se le da una segunda oportunidad a aquellos elementos con el bit de acceso a 1	

Hiperpaginación o thrashing

Ocurre cuando la suma de las páginas usadas por todos los procesos es mayor que el número de tramas físicas disponibles en el sistema. Hay páginas del Working Set de varios procesos que residen en el swap. Cuando en un proceso A se produce un fallo de página, la página reemplazada también forma parte del WS de otro proceso B. A su vez, el proceso B puede necesitar la página que ha sido reemplazada, por lo que la CPU puede terminar ociosa la mayor parte del tiempo (uso del 0%). La solución es controlar el grado de multiprogramación y suspender los procesos para liberar sus marcos

Ejemplos

Memoria virtual en Linux

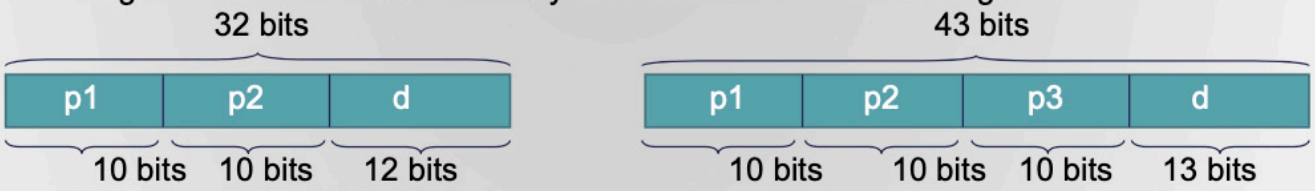
Algoritmo de reemplazo con clock y LFU. El contador mide la juventud (frecuencia de acceso) de la página. En cada fallo de página se dividen todos los contadores y encuentran el mínimo. Se selecciona aquella con menos juventud como víctima

Las páginas se almacenan en una partición conocida como *swap*, sin ficheros. La tabla de páginas se mantiene cargada en memoria.

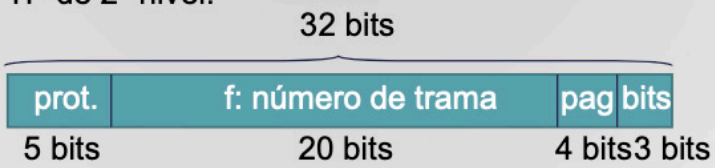
Memoria virtual Windows XP

Utiliza el algoritmo de reemplazo Working Set. A cada proceso se le asignan inicialmente 50 marcos. Cada cierto tiempo a los procesos se le quita un marco. Si dicho proceso no genera fallo, se decrementa el WS de dicho proceso. Si el proceso supera la tasa de fallos umbral, entonces se incrementa su WS. Los reemplazos de página ocurren dentro de los propios marcos del WS, sin acceder a los de otro proceso. El reemplazo se decide usando FIFO

- **Memoria Virtual en Windows XP:**
  - Paginación en 2 niveles en IA32 y en 3 en IA64. Direcciones lógicas:



  - Entradas de la TP de 2º nivel:

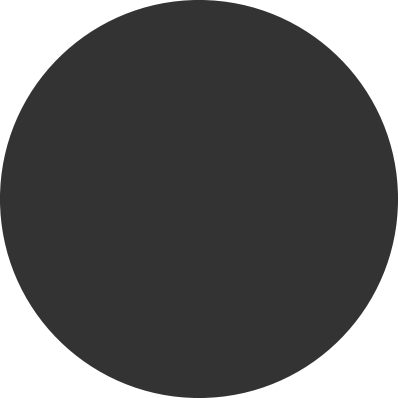


    - Prot: bits de protección (*read-only*, *read-write*, …)
    - Pag: indica el archivo de *swap* que respalda la página
    - Bits: T (en transición), D (*dirty*) y P (presencia)
    - Cuando P=0 los 20 bits del campo *f* apuntan a la página en el *swap*
  - Precarga: en caso de fallo → trae páginas adicionales

Ejercicios

Nombre	Descripción	Fórmula
Offset	Desplazamiento en el interior de una página. Se corresponde con los últimos bits de la dirección	Bits finales
Tamaño de página	Máximo offset de una página	$2^{Nbits\ Offset}$
Tamaño de marco	Máximo offset de una página	$2^{Nbits\ Offset}$
Número de marcos		
Número de páginas		
Dirección física	Dirección que apunta a un determinado bloque de memoria	
Dirección proceso	Dirección que el cargador asigna a una determinada sección de memoria del programa	
Dirección de programa	Dirección que el enlazador asigna a un código compilado	
Entrada de tabla de páginas	Incluye los flags y el número de marco	$b_{flags} + b_{marco}$

- Niveles de multipaginación



    - offset: $log_2(\frac{Tam\ bloque}{Direccionamiento})$
    - Niveles intermedios: $log_2(N_{entradas})$
    - Nivel superior: $N_{bL} - N_{b1} - N_{b2} \dots$
- Direccionamiento en páginas
    - Invalida: Cuando el bit válido es 0
    - Fallo TLB: Si dicha dirección lógica no está en la tabla de la TLB
    - Fallo página: presencia 0
- Porcentaje de entradas no usadas de la tabla de páginas: relación entre el número de páginas no usadas y el total de páginas. No confundir con sus bits de indicación

Otros

- MMU: Memory Management Unit. Recibe en cada cambio de contexto la tabla de página que debe utilizar junto con el límite superior e inferior de direcciones
- TP: Tabla de páginas
- PTBR: Registro ubicado dentro de la PCB que indica el punto de inicio de la tabla de páginas de ese proceso
- Bit de presencia: Un bit habilitado en la tabla de páginas cuando dicha página está cargada en la memoria principal
- Cuando se produce un fallo de página, se lanza una interrupción/excepción
- Working Set: Calcula el número de marcos óptimo que necesita un proceso utilizando una ventana de tiempo