

Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

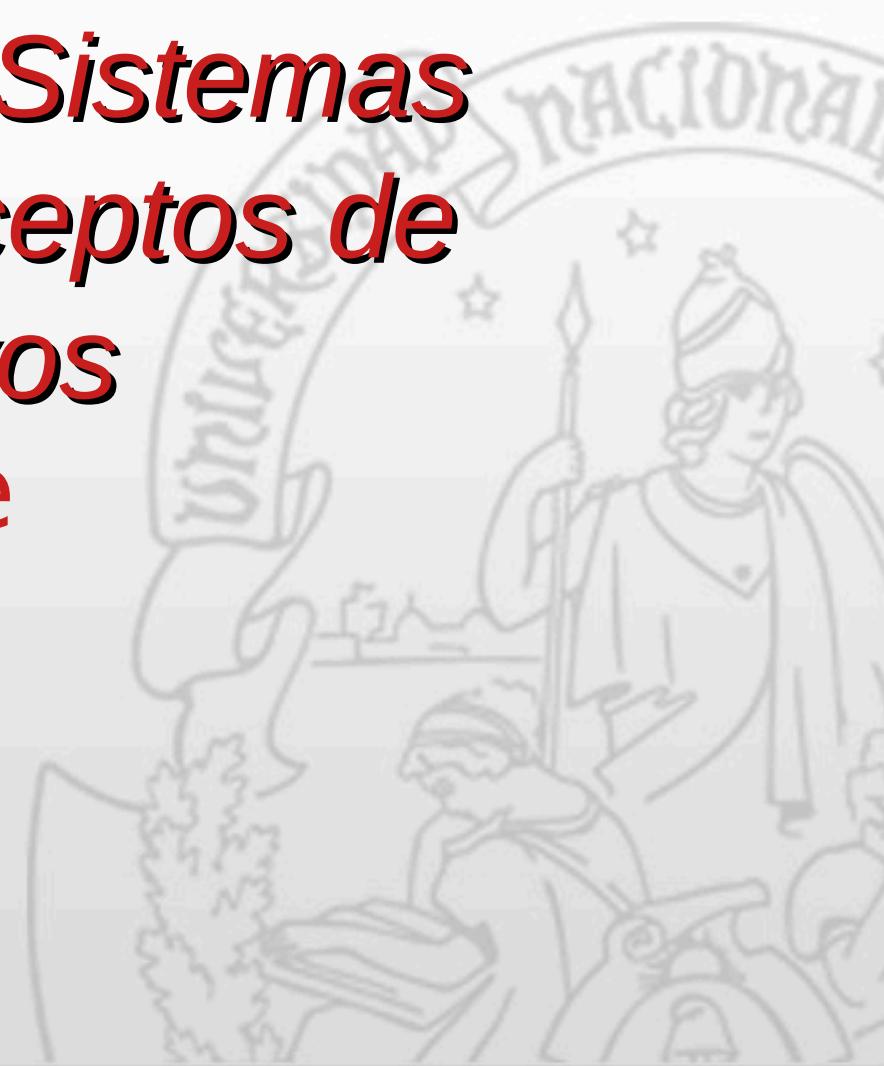
Administración de Memoria - I

Profesores:

Lía Molinari

Juan Pablo Pérez

Nicolás del Rio



- Versión: Septiembre 2019
- Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Paginación, Segmentación, Fragmentación

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Memoria

- ❑ La organización y administración de la “memoria principal” es uno de los factores más importantes en el diseño de los S. O.
- ❑ Los programas y datos deben estar en el almacenamiento principal para:
 - ❑ Poderlos ejecutar.
 - ❑ Referenciarlos directamente.



Memoria (cont.)

- ❑ El SO debe:
 - ❑ Llevar un registro de las partes de memoria que se están utilizando y de aquellas que no.
 - ❑ Asignar espacio en memoria principal a los procesos cuando estos la necesitan.
 - ❑ Libera espacio de memoria asignada a procesos que han terminado.
- ❑ Se espera de un S.O. un uso eficiente de la memoria con el fin de alojar el mayor número de procesos



Memoria

(cont.)

□ El S.O. debe:

- Lograr que el programador se abstraiga de la alocación de los programas
- Brindar seguridad entre los procesos para que unos no accedan a secciones privadas de otros
- Brindar la posibilidad de acceso compartido a determinadas secciones de la memoria (librerías, código en común, etc.)
- Garantizar la performance del sistema



Administración de Memoria

- División Lógica de la Memoria Física para alojar múltiples procesos
 - Garantizando protección
 - Depende del mecanismo provisto por el HW
- Asignación eficiente
 - Contener el mayor numero de procesos para garantizar el mayor uso de la CPU por los mismos



Requisitos

Reubicación

- ✓ El programador no debe ocuparse de conocer donde será colocado en la Memoria RAM
- ✓ Mientras un proceso se ejecuta, puede ser sacado y traído a la memoria (swap) y, posiblemente, colocarse en diferentes direcciones.
- ✓ Las referencias a la memoria se deben “traducir” según ubicación actual del proceso.



Requisitos (cont.).

Protección

- ✓ Los procesos NO deben referenciar – acceder - a direcciones de memoria de otros procesos
 - Salvo que tengan permiso
- ✓ El chequeo se debe realizar durante la ejecución:
 - ◆ NO es posible anticipar todas las referencias a memoria que un proceso puede realizar.



Requisitos (cont.).

Compartición

- ✓ Permitir que varios procesos accedan a la misma porción de memoria.
 - ◆ Ej: Rutinas comunes, librerías, espacios explícitamente compartidos, etc.
- ✓ Permite un mejor uso - aprovechamiento - de la memoria RAM, evitando copias innecesarias (repetidas) de instrucciones

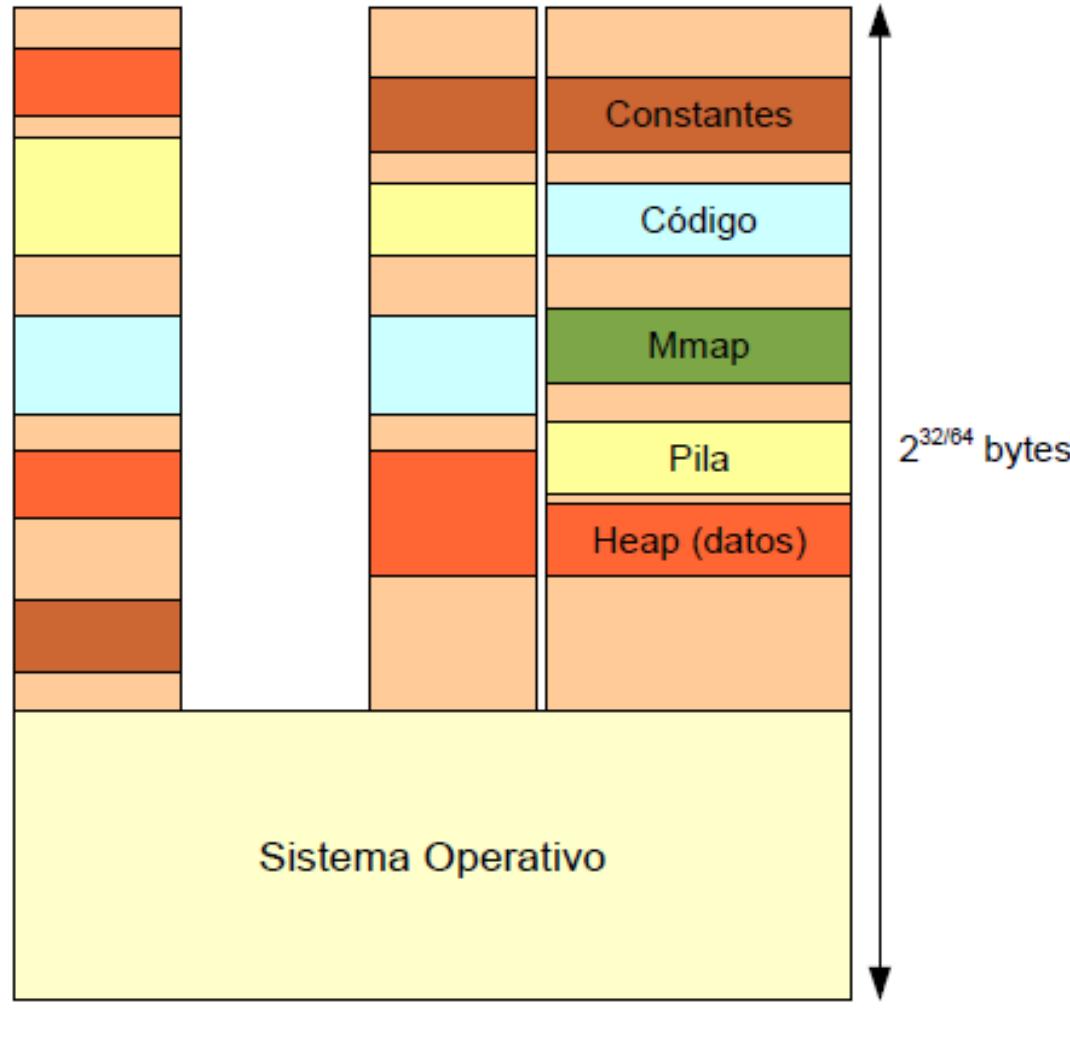


Abstracción - Espacio de Direcciones

- Rango de direcciones (a memoria) posibles que un proceso puede utilizar para direccionar sus instrucciones y datos.
- El tamaño depende de la Arquitectura del Procesador
 - ✓ 32 bits: $0 .. 2^{32} - 1$
 - ✓ 64 bits: $0 .. 2^{64} - 1$
- Es independiente de la ubicación “real” del proceso en la Memoria RAM



Abstracción -Espacio de Direcciones (cont.)



Direcciones

Lógicas

- ✓ Referencia a una localidad de memoria independiente de la asignación actual de los datos en la memoria.
- ✓ Representa una dirección en el “Espacio de Direcciones del Proceso”

Físicas

- ✓ Referencia una localidad en la Memoria Física (RAM)
 - Dirección absoluta

En caso de usar direcciones Lógicas, es necesaria algún tipo de conversión a direcciones Físicas.



Conversión de Direcciones

Una forma simple de hacer esto es utilizando registros auxiliares

Registro Base

- ✓ Dirección de comienzo del Espacio de Direcciones del proceso en la RAM

Registro Límite

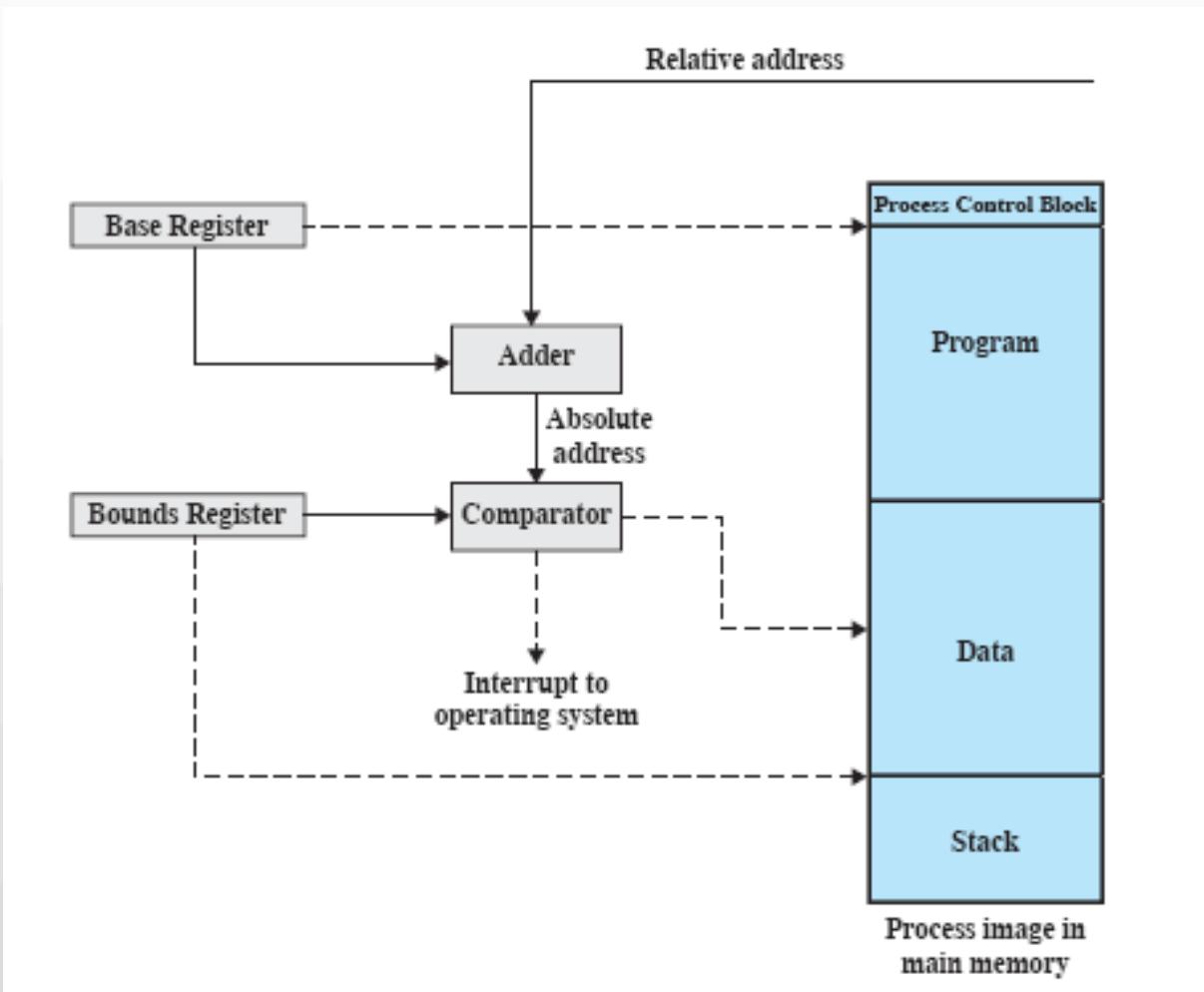
- ✓ Dirección final del proceso o medida del proceso
 - Tamaño de su Espacio de Direcciones

Ambos valores se fijan cuando el espacio de direcciones del proceso es cargado a memoria.

Varían entre procesos (Context Switch)



Direcciones (cont.)



Dir. Lógicas vs. Físicas

- Si la CPU trabaja con direcciones lógicas, para acceder a memoria principal, se deben transformar en direcciones físicas.
 - Resolución de direcciones (address-binding): transformar la dirección lógica en la dirección física correspondiente
- Resolución en momento de compilación (Archivos .com de DOS) y en tiempo de carga
 - ✓ Direcciones Lógicas y Físicas son idénticas
 - ✓ Para reubicar un proceso es necesario recompilarlo o recargarlo.



Dir. Lógicas vs. Físicas

Resolución en tiempo de ejecución

- ✓ Direcciones Lógicas y Físicas son diferentes
- ✓ Direcciones Lógicas son llamadas “Direcciones Virtuales”
- ✓ La reubicación se puede realizar fácilmente
- ✓ El mapeo entre “Virtuales” y “Físicas” es realizado por hardware
 - Memory Management Unit (MMU)



Memory Management Unit (MMU)

Dispositivo de Hardware que mapea direcciones virtuales a físicas

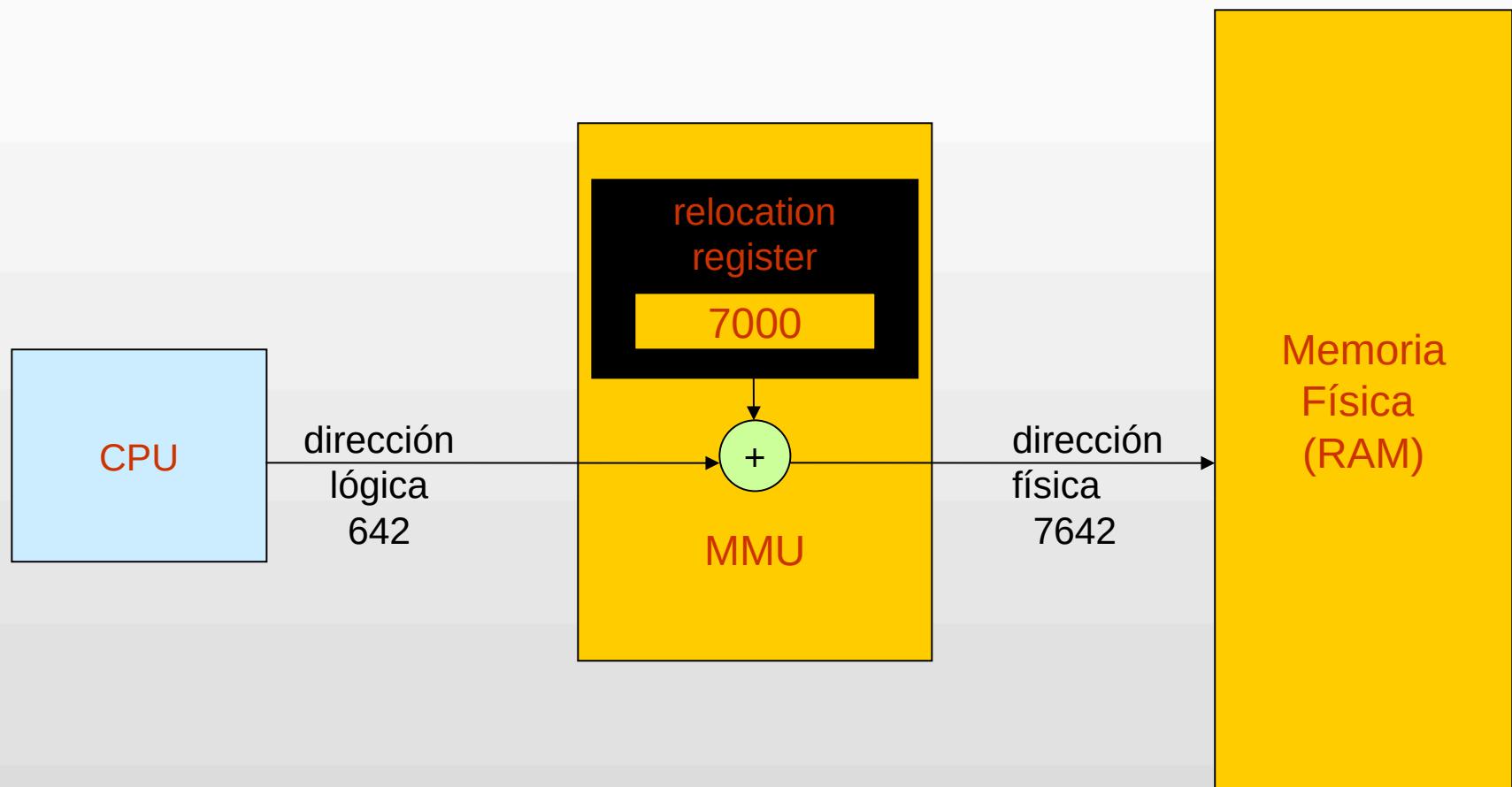
- ✓ Es parte del Procesador
- ✓ Re-programar el MMU es una operación privilegiada
 - solo puede ser realizada en Kernel Mode

El valor en el “registro de realocación” es sumado a cada dirección generada por el proceso de usuario al momento de acceder a la memoria.

- ✓ Los procesos nunca usan direcciones físicas



MMU



Mecanismos de asignación de memoria

- Particiones Fijas: El primer esquema implementado
 - ✓ La memoria se divide en particiones o regiones de tamaño Fijo (pueden ser todas del mismo tamaño o no)
 - ✓ Alojan un proceso cada una
 - ✓ Cada proceso se coloca de acuerdo a algún criterio (First Fit, Best Fit, Worst Fit, Next Fit) en alguna partición

- Particiones dinámicas: La evolución del esquema anterior
 - ✓ Las particiones varían en tamaño y en número
 - ✓ Alojan un proceso cada una
 - ✓ Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso

¿Qué problemas se generan en cada caso?



Fragmentación

La fragmentación se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua

Fragmentación Interna:

- ✓ Se produce en el esquema de particiones Fijas
- ✓ Es la porción de la partición que queda sin utilizar

Fragmentación Externa:

- ✓ Se produce en el esquema de particiones dinámicas
- ✓ Son huecos que van quedando en la memoria a medida que los procesos finalizan
- ✓ Al no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alojar un proceso, pero que no la podamos utilizar
- ✓ Para solucionar el problema se puede acudir a la compactación, pero es muy costosa



Problemas del esquema

- El esquema de Registro Base + Límite presenta problemas:
 - Necesidad de almacenar el Espacio de Direcciones de forma continua en la Memoria Física
 - Los primeros SO definían particiones fijas de memoria, luego evolucionaron a particiones dinámicas
 - Fragmentación
 - Mantener “partes” del proceso que no son necesarias
 - Los esquemas de particiones fijas y dinámicas no se usan hoy en día

Solución:

- Paginación
- Segmentación



Paginación

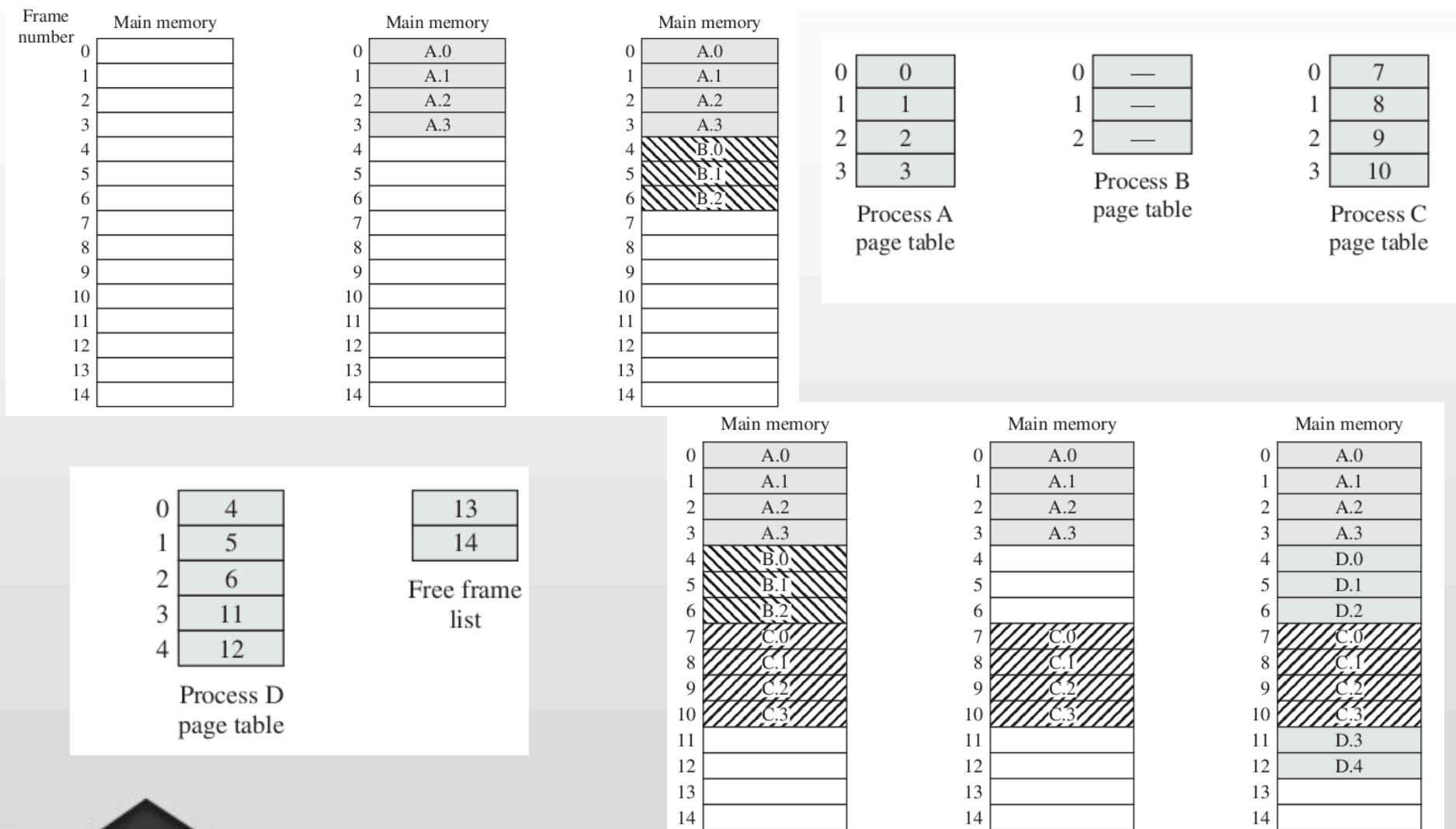
- Memoria Física es dividida lógicamente en pequeños trozos de igual tamaño → **Marcos**
- Memoria Lógica (espacio de direcciones) es dividido en trozos de igual tamaño que los marcos → **Paginas**
- El SO debe mantener una tabla de paginas por cada proceso, donde cada entrada contiene (entre otras) el **Marco** en la que se coloca cada pagina.
- La dirección lógica se interpreta como:
 - un numero de pagina y un desplazamiento dentro de la misma.



Paginación - Ejemplo I



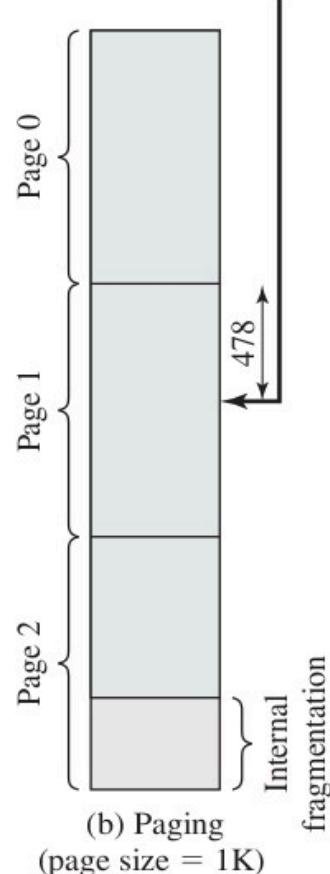
Paginación - Ejemplo II



Paginación - Direcciones Lógicas

Logical address =
Page# = 1, Offset = 478

000001|0111011110



16-bit logical address

6-bit page # 10-bit offset

0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 0

0 000101
1 000110
2 011001

Process
page table

16-bit physical address

(a) Paging



Traducción de direcciones

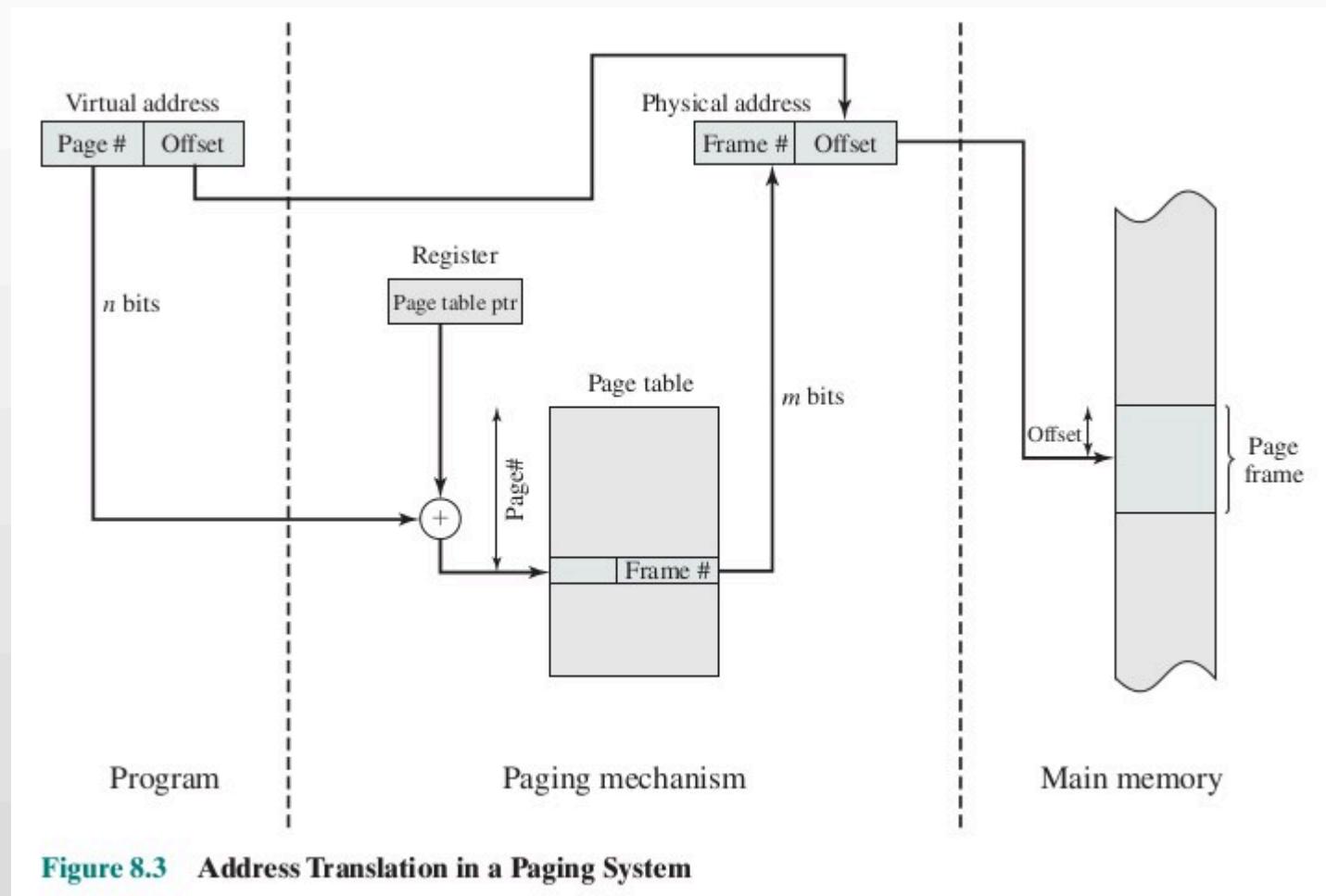


Figure 8.3 Address Translation in a Paging System

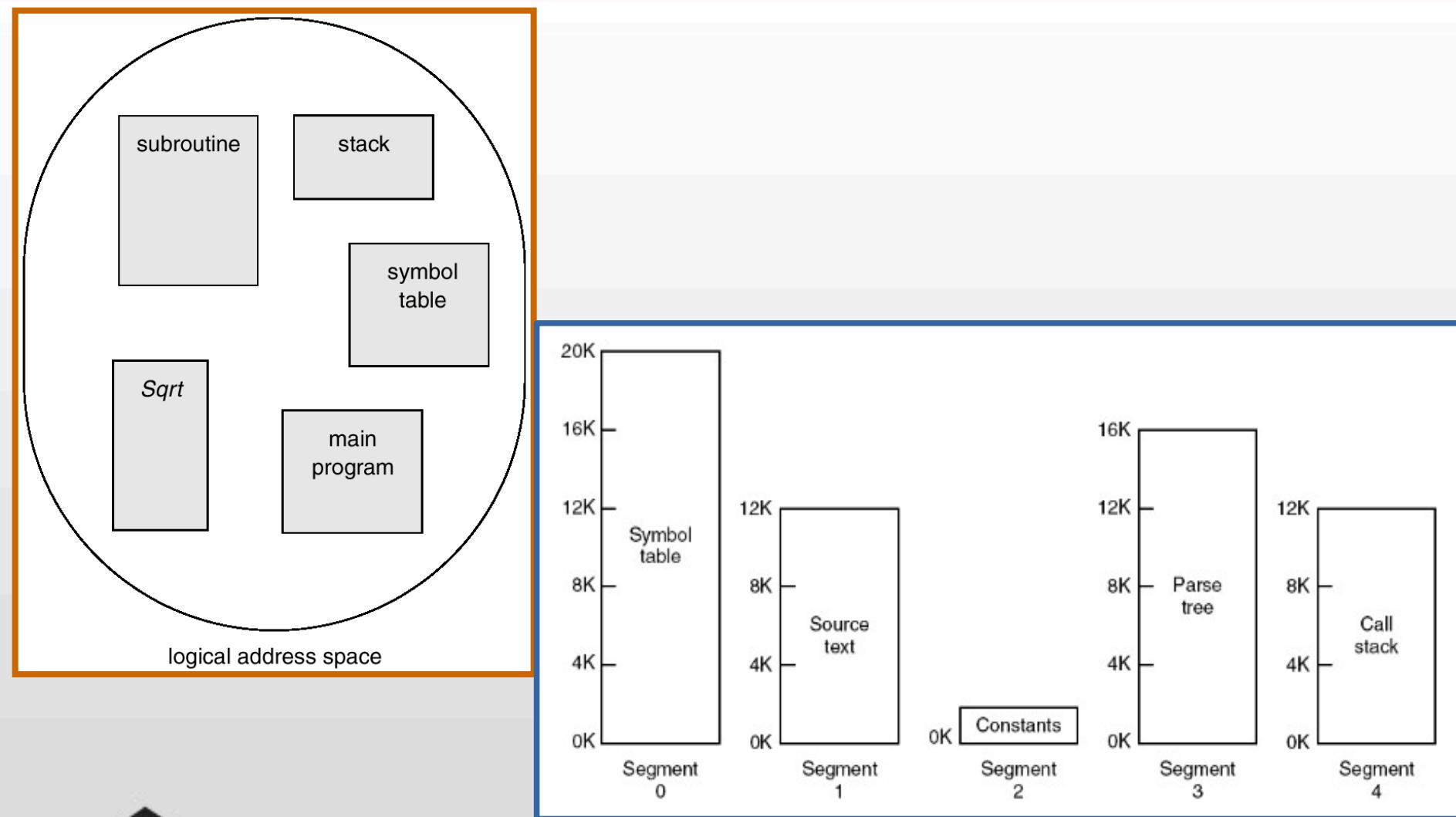


Segmentación

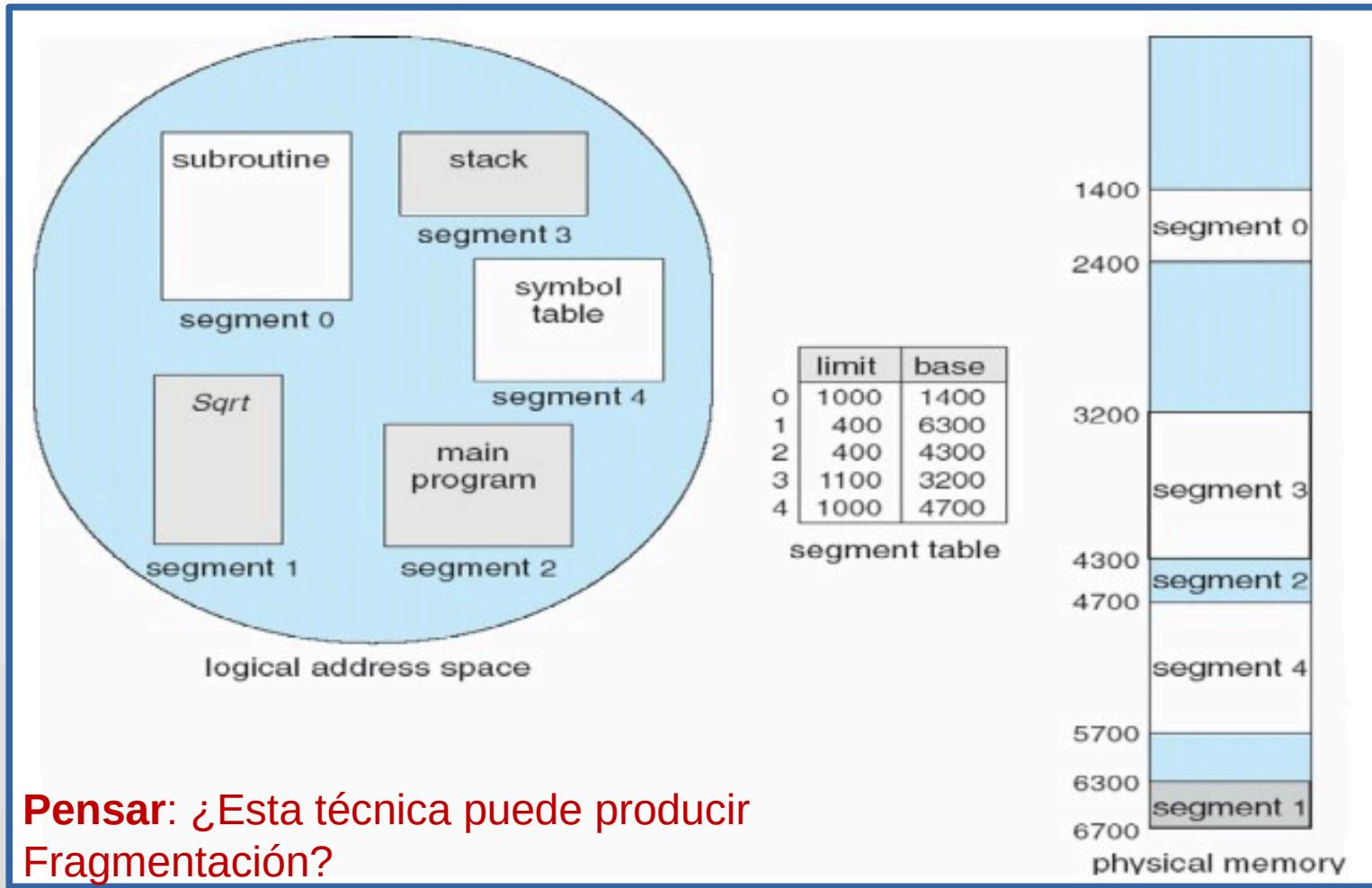
- Esquema que se asemeja a la “visión del usuario”. El programa se divide en partes/secciones
- Un programa es una colección de segmentos. Un segmento es una unidad lógica como:
 - Programa Principal, Procedimientos y Funciones, variables locales y globales, stack, etc.
- Puede causar Fragmentación



Programa desde la visión del usuario



Ejemplo de Segmentación



Pensar: ¿Esta técnica puede producir Fragmentación?



Segmentación (cont.)

- ✓ Todos los segmentos de un programa pueden no tener el mismo tamaño (código, datos, rutinas).
- ✓ Las direcciones Lógicas consisten en 2 partes:
 - ✓ Selector de Segmento
 - ✓ Desplazamiento dentro del segmento



Segmentación (cont.) - Arquitectura

Tabla de Segmentos

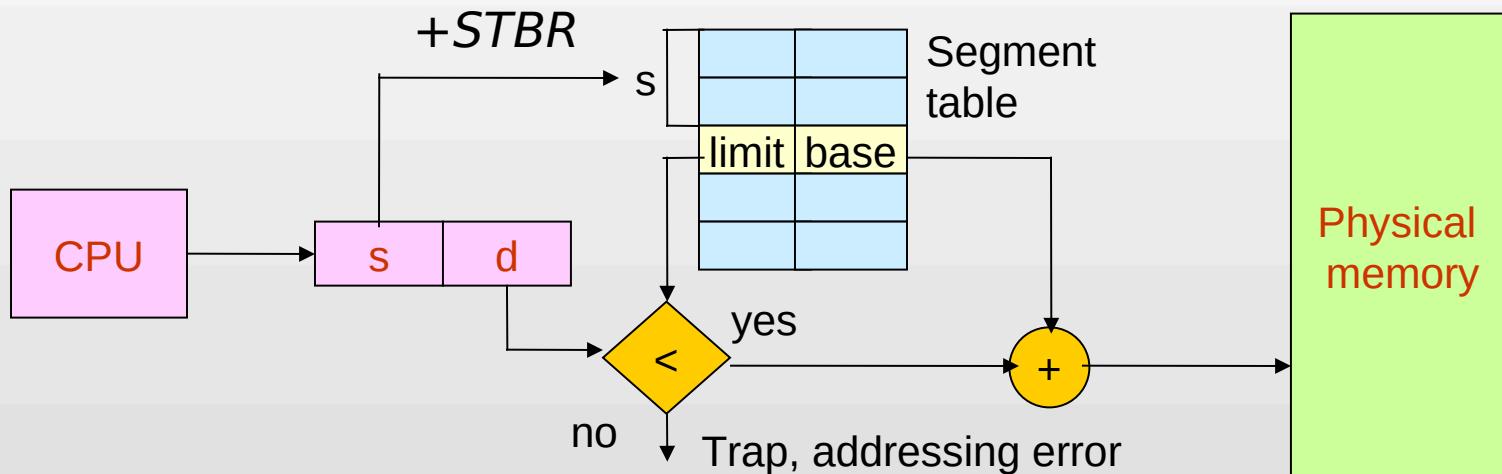
- ✓ Permite mapear la dirección lógica en física. Cada entrada contiene:
 - ◆ Base: Dirección física de comienzo del segmento
 - ◆ Limit: Longitud del Segmento

Segment-table base register (*STBR*): apunta a la ubicación de la tabla de segmentos.

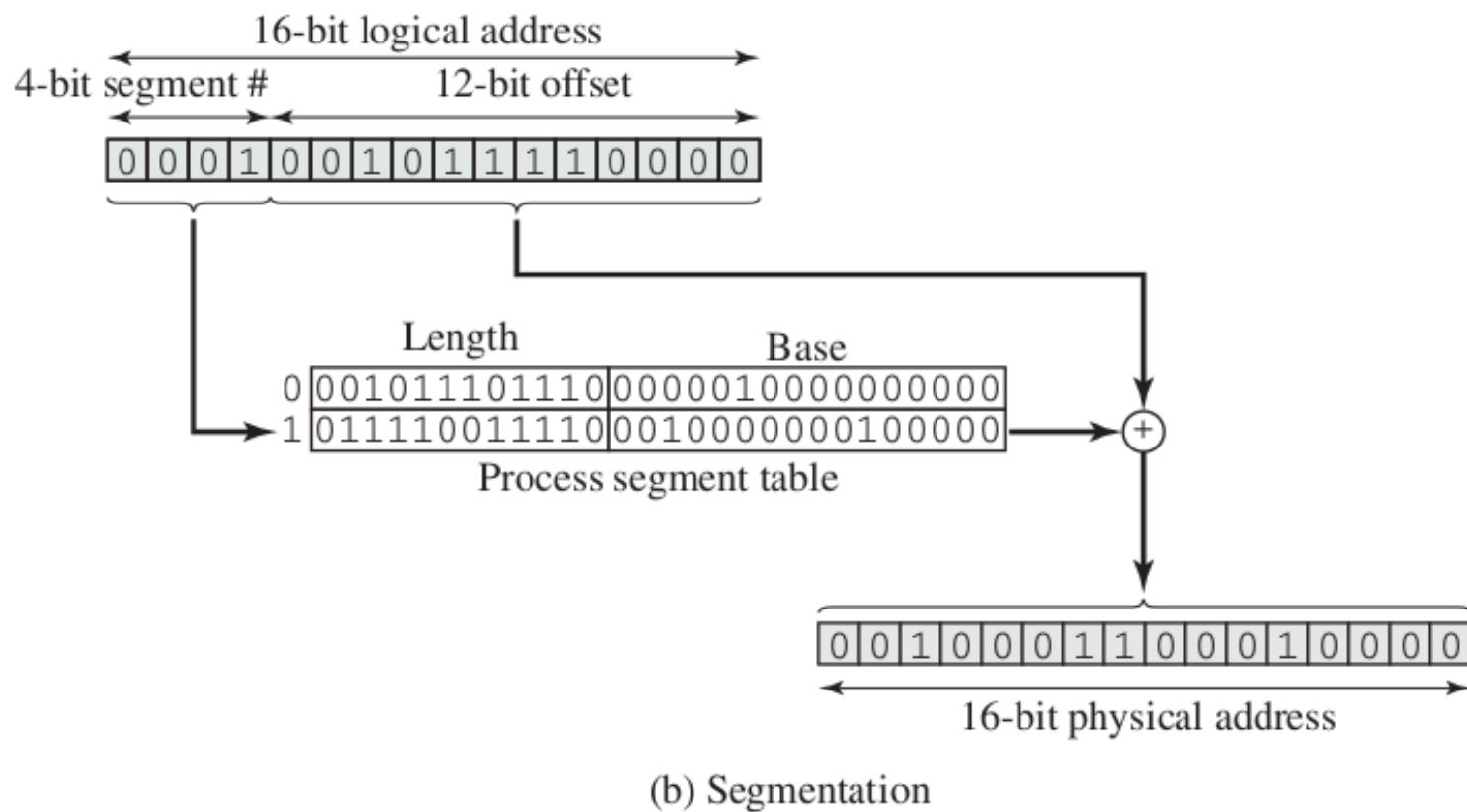
Segment-table length register (*STLR*) : cantidad de segmentos de un programa



Segmentación (cont.)



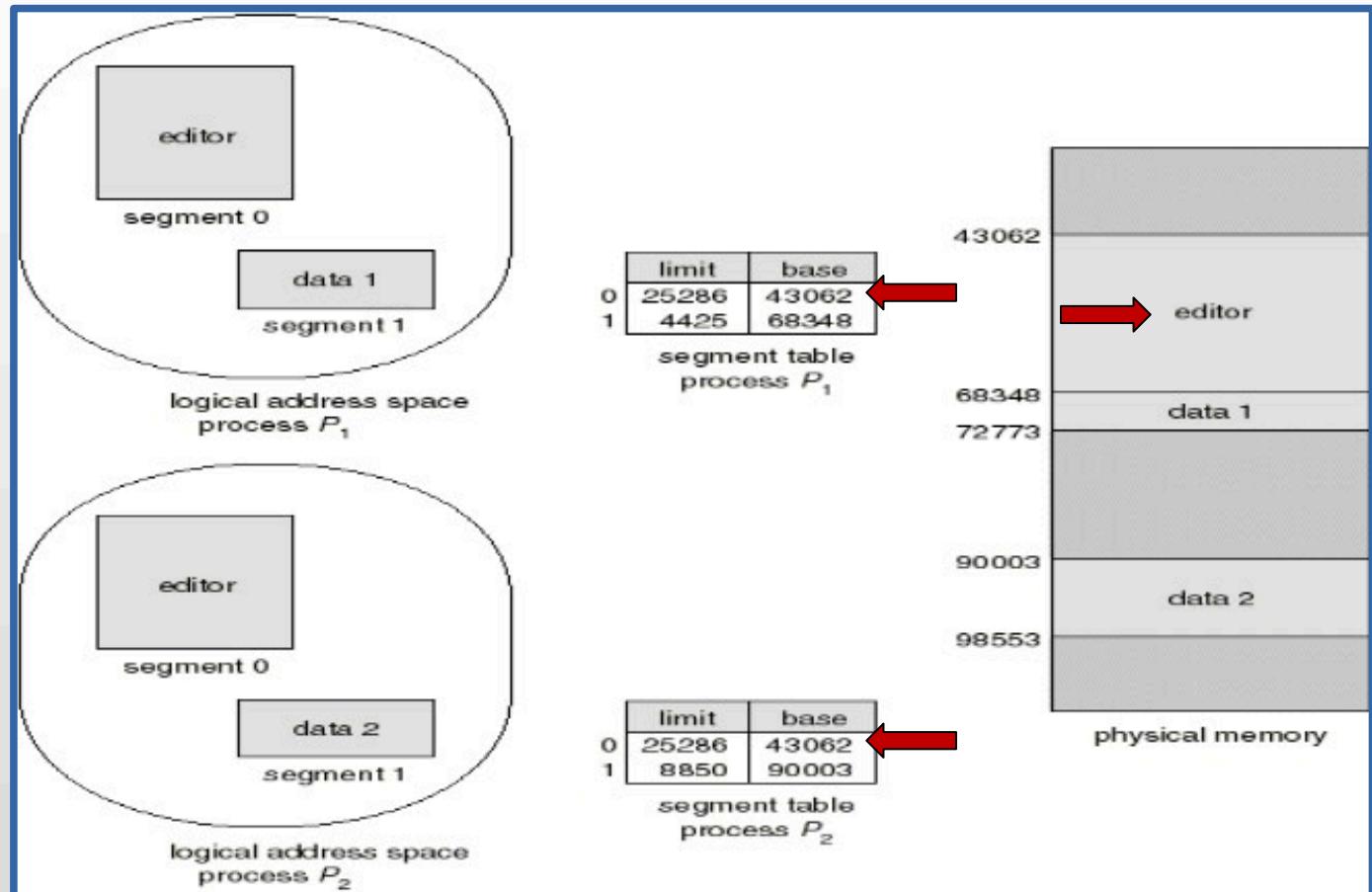
Segmentación - Direcciones (cont.)



Ventajas sobre Segmentación

Compartir

Proteger



Segmentación Paginada

La paginación

- ✓ Transparente al programador
- ✓ Elimina Fragmentación externa.

Segmentación

- ✓ Es visible al programador
- ✓ Facilita modularidad, estructuras de datos grandes y da mejor soporte a la compartición y protección

Segmentación Paginada: Cada segmento es dividido en páginas de tamaño fijo.



Segmentación Paginada (cont.)

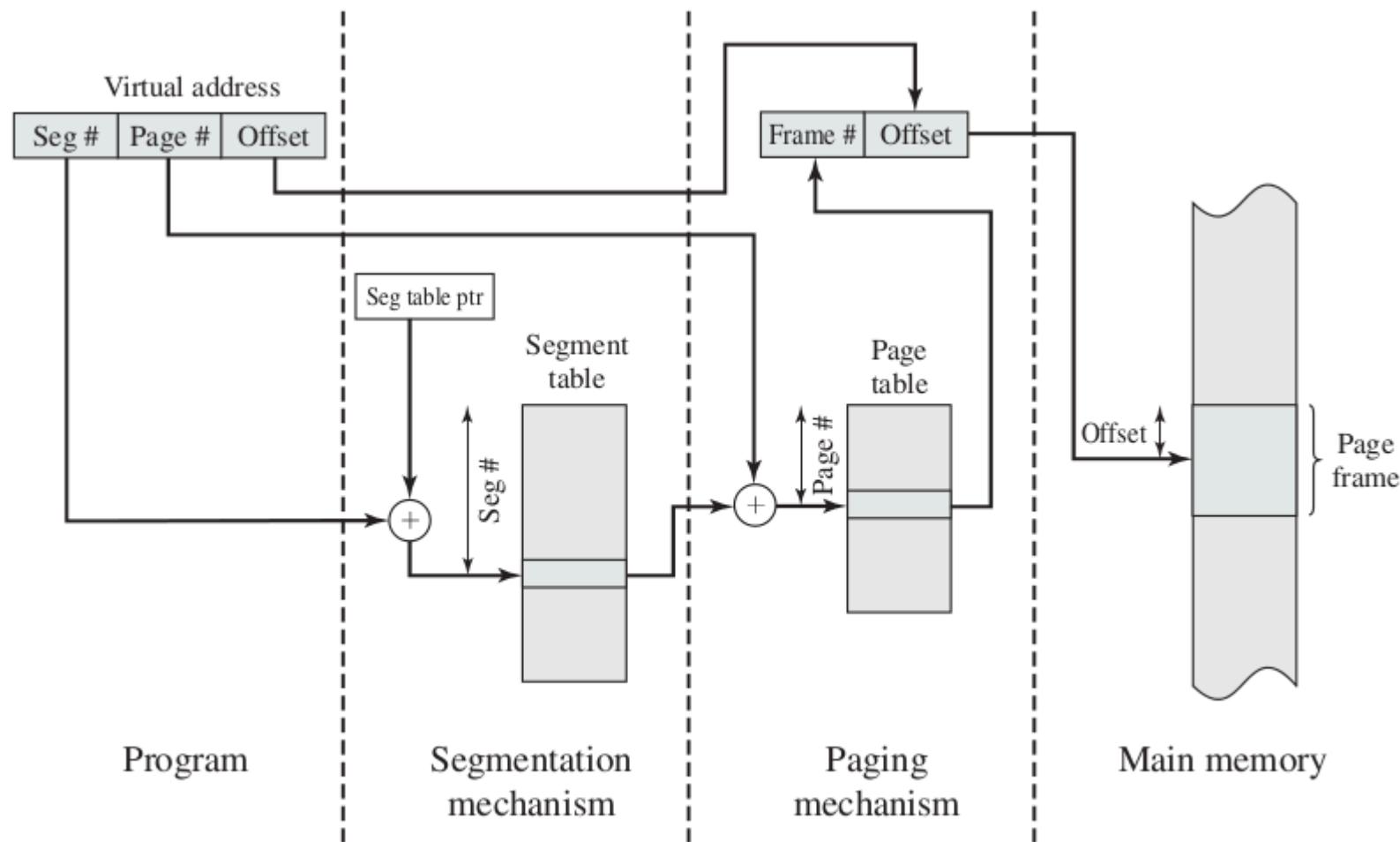
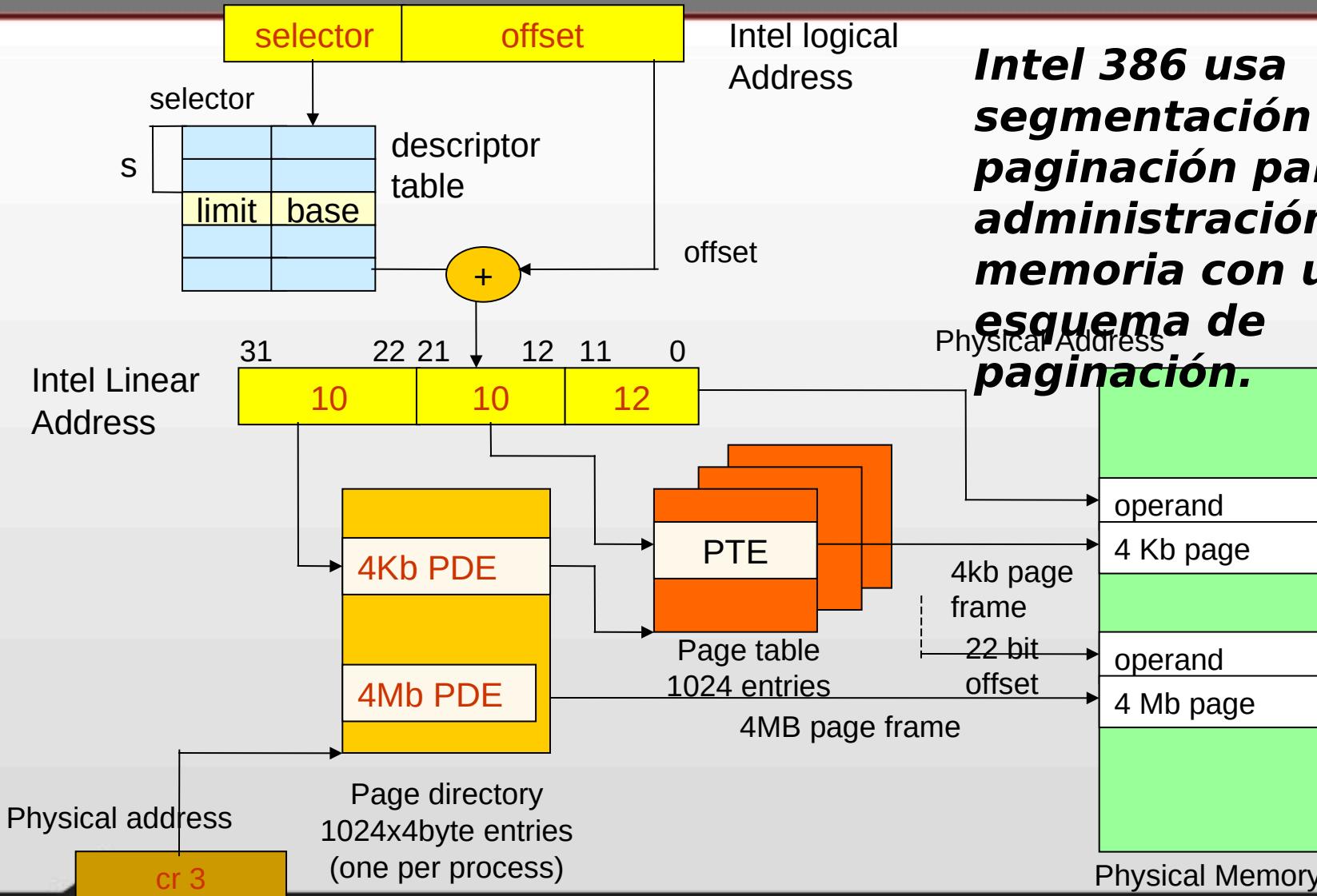


Figure 8.13 Address Translation in a Segmentation/Paging System



Intel x386



Intel 386 usa segmentación con paginación para la administración de la memoria con un doble esquema de paginación.



Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de
Memoria - II



Versión: Octubre 2022

Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Paginación, Memoria Virtual, Tablas de Páginas

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Hasta ahora

- ✓ Con paginación vimos que el espacio de direcciones de un proceso no necesariamente debe estar “contiguo” en la memoria para poder ejecutarse
- ✓ El hardware traduce direcciones lógicas a direcciones físicas utilizando las tablas de páginas que el SO administra



Motivación para Memoria Virtual

- Podemos pensar también que, no todo el espacio de direcciones del proceso se necesitó en todo momento:
 - ✓ Rutinas o Librerías que se ejecutan una única vez (o nunca)
 - ✓ Partes del programa que no vuelven a ejecutarse
 - ✓ Regiones de memoria alocadas dinámicamente y luego liberadas
 - ✓ Etc.
- ✓ No hay necesidad que la totalidad la imagen del proceso sea cargada en memoria



Como se puede trabajar...

- ✓ El SO puede traer a memoria las “piezas” de un proceso a medida que éste las necesita.
- ✓ Definiremos como “**Conjunto Residente**” a la porción del espacio de direcciones del proceso que se encuentra en memoria.
 - ✓ Alguna bibliografía lo llama “Working Set”
- ✓ Con el apoyo del HW:
 - ✓ Se detecta cuando se necesita una porción del proceso que no está en su Conjunto Residente
 - ✓ Se debe cargar en memoria dicha porción para continuar con la ejecución.



Ventajas

- Más procesos pueden ser mantenidos en memoria.
 - ✓ Sólo son cargadas algunas secciones de cada proceso.
 - ✓ Con más procesos en memoria principal es más probable que existan más procesos Ready
- Un proceso puede ser más grande que la memoria Principal
 - ✓ El usuario no se debe preocupar por el tamaño de sus programas
 - ✓ La limitación la impone el HW y el bus de direcciones.



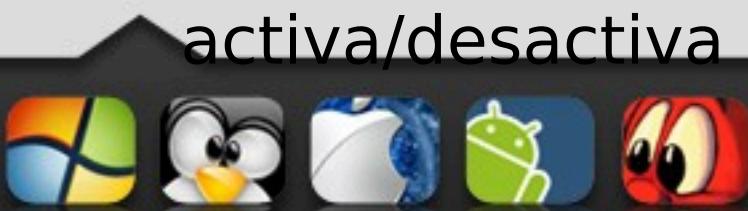
¿Que se necesita para Memoria Virtual?

- El hardware debe soportar paginación por demanda (y/o segmentación por demanda)
- Un dispositivo de memoria secundaria (disco) que dé el apoyo para almacenar las secciones del proceso que no están en Memoria Principal (área de intercambio)
- El SO debe ser capaz de manejar el movimiento de las páginas (o segmentos) entre la memoria principal y la secundaria.

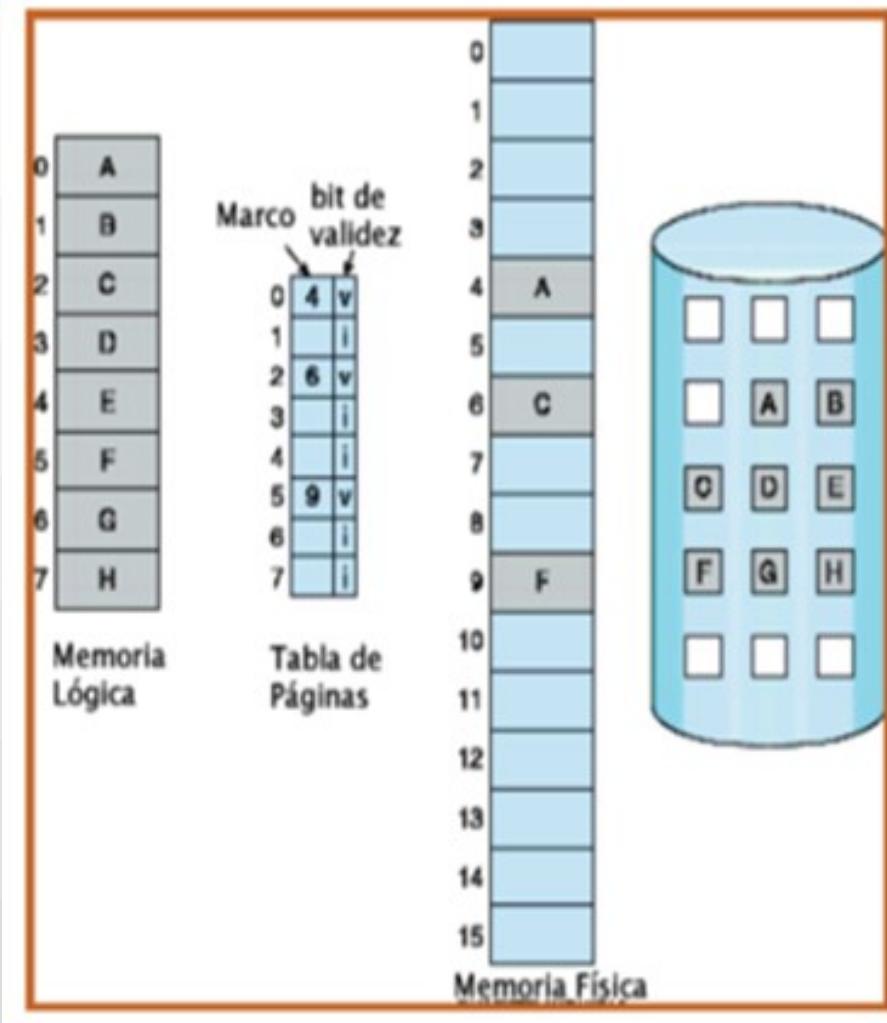


Memoria Virtual con Paginación

- ✓ Cada proceso tiene su tabla de páginas
- ✓ Cada entrada en la tabla referencia al frame o marco en el que se encuentra la página en la memoria principal
- ✓ Cada entrada en la tabla de páginas tiene bits de control (entre otros):
 - ✓ **Bit V:** Indica si la página está en memoria (lo activa/desactiva el SO, lo consulta el HW)
 - ✓ **Bit M:** Indica si la página fue modificada. Si se modificó, en algún momento, se deben reflejar los cambios en Memoria Secundaria (lo activa/desactiva el HW, lo consulta el SO)



Memoria Virtual con Paginación

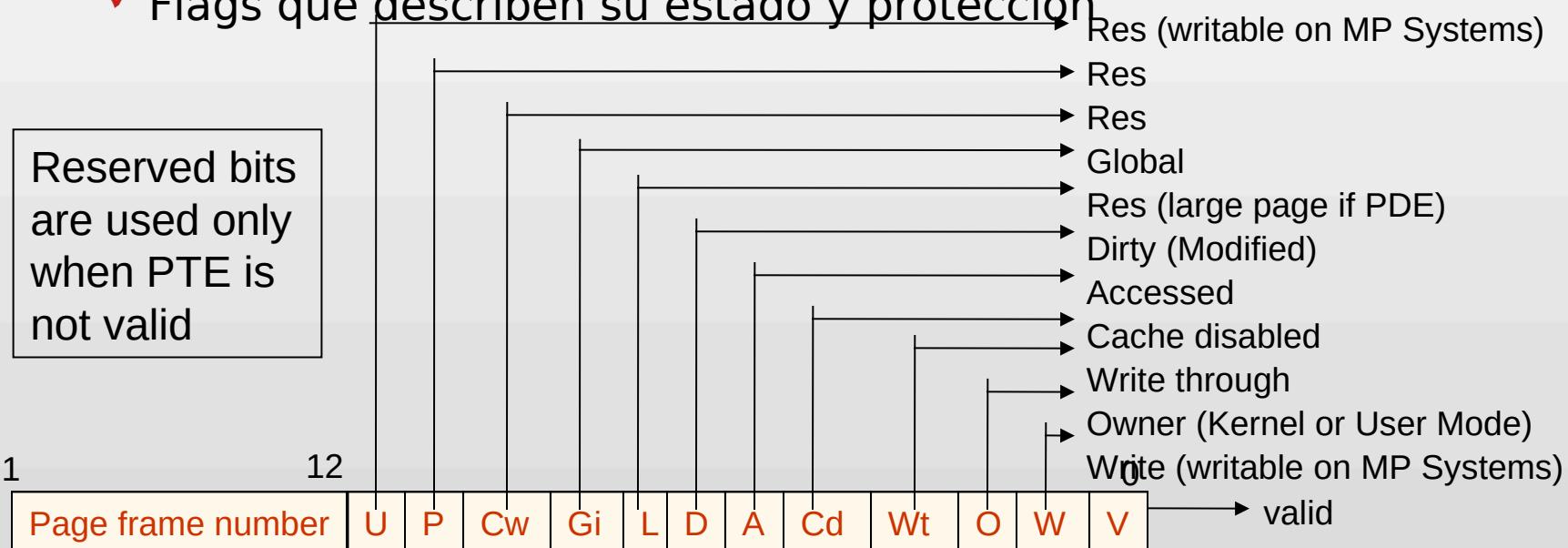


Entrada en la Tabla de páginas de x86 (32 bits)

El HW define el formato de la tabla de páginas y el SO se adapta a él

Una entrada válida tiene:

- ✓ Bit V = 1
- ✓ Page Frame Number (PFN) – Marco de memoria asociado
- ✓ Flags que describen su estado y protección



Fallo de páginas (Page Fault)

- ✓ Ocurre cuando el proceso intenta usar una dirección que está en una página que no se encuentra en la memoria principal. Bit V=0 (también marcado con i = inválido)
 - ✓ La página no se encuentra en su conjunto residente
- ✓ El HW detecta la situación y genera un trap al S.O.
- ✓ El S.O. Podrá colocar al proceso en estado de “Blocked” (espera) mientras gestiona que la página que se necesite se cargue.



Fallo de páginas (cont.)

- ✓ El S.O. busca un “Frame o Marco Libre” en la memoria y genera una operación de E/S al disco para copiar en dicho Frame la página del proceso que se necesita utilizar.
- ✓ El SO puede asignarle la CPU a otro proceso mientras se completa la E/S
 - ✓ La E/S se realizará y avisará mediante interrupción su finalización.

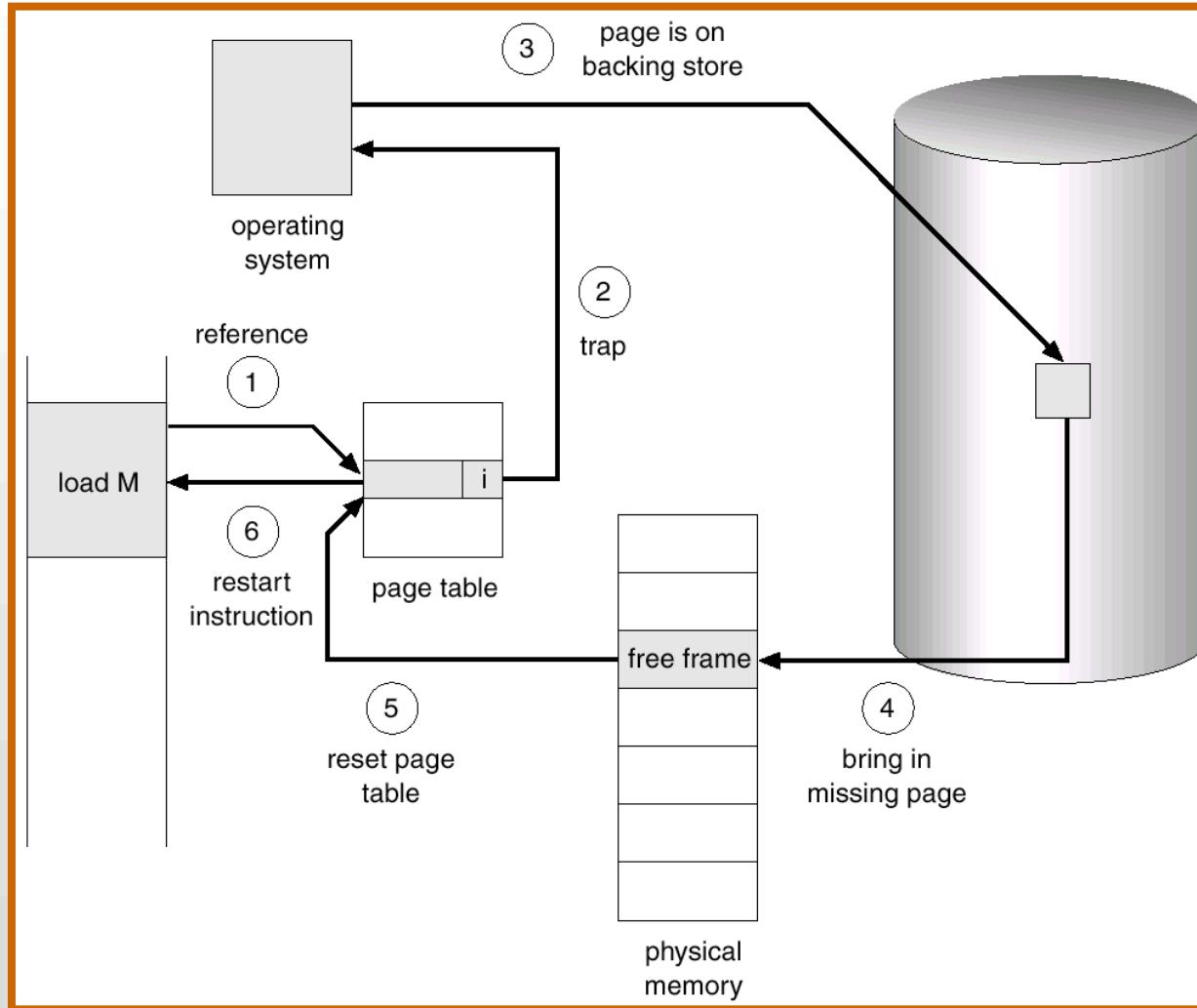


Fallo de páginas (cont.)

- ✓ Cuando la operación de E/S finaliza, se notifica al SO y este:
 - ✓ Actualiza la tabla de páginas del proceso
 - ◆ Coloca el Bit V en 1 en la página en cuestión
 - ◆ Coloca la dirección base del Frame donde se colocó la página
 - ✓ El proceso que generó el Fallo de Página vuelve a estado de Ready (listo)
 - ✓ Cuando el proceso se ejecute, se volverá a ejecutar la instrucción que antes generó el fallo de página



Fallo de páginas (cont.)



Fallo de páginas (cont.)

- La técnica de paginación intenta alocar la mayor cantidad de páginas necesarias posibles
- Cada vez que hay que alocar una página en un marco, se produce un fallo de página
- ¿Qué sucede si es necesario alocar una página y ya no hay marcos disponibles?
- Se debe seleccionar una página víctima, para lo cual existen diversos algoritmos (FIFO, Óptimo, LRU, etc.)
- ¿Cuál es el mejor algoritmo?:
- El que seleccione como página víctima aquella que no vaya a ser referenciada en un futuro próximo



Performance

Si los page faults son excesivos, la performance del sistema decae

Tasa de Page Faults $0 \leq p \leq 1$

- ✓ Si $p = 0$ no hay page faults

- ✓ Si $p = 1$, cada a memoria genera un page fault

Effective Access Time (EAT)

$$\begin{aligned} EAT = & (1 - p) \times \text{memory access} \\ & + p \times (\cancel{\text{page_fault_overhead}} + \\ & \quad [\text{swap_page_out}] + \\ & \quad \cancel{\text{swap_page_in}} + \\ & \quad \cancel{\text{restart_overhead}}) \end{aligned}$$

Podría ocurrir que no haya marcos disponibles, con lo cual habrá que descargar uno para lograr espacio para la nueva página entrante



Tabla de Páginas

- ✓ Cada proceso tiene su tabla de páginas
- ✓ El tamaño de la tabla de páginas depende del espacio de direcciones del proceso.
- ✓ Puede alcanzar un tamaño considerable



Tabla de páginas (cont.)

Formas de organizar:

- ✓ Tabla de 1 nivel: Tabla única lineal
- ✓ Tabla de 2 niveles (o más, multinivel)
- ✓ Tabla invertida: Hashing

La forma de organizarla depende del HW subyacente



Tabla de 1 nivel - 32 bits

- ✓ Direcciones de 32bit



- ✓ Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{20} (1.048.576)
- ✓ El tamaño de cada página es de 4KB (2^{12})
- ✓ El tamaño de cada PTE es de 4 bytes
 - ✓ Cantidad de PTEs que entran en un marco: $4\text{KB}/4\text{B} = 2^{10}$
- ✓ Tamaño de tabla de páginas
 - ◆ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = $2^{20}/2^{10} = 2^{10}$
 - ◆ Tamaño tabla de páginas del proceso:
 $2^{10} * 4\text{bytes} = \mathbf{4MB \ por \ proceso}$



Tabla de 1 nivel - 64 bits

Direcciones de 64bits

52 bits

12 bits

Numero de página

Desplazamiento

Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{52}
- ✓ El tamaño de cada página es de 4KB
- ✓ El tamaño de cada PTE es de 4 bytes
 - ◆ Cantidad de PTEs que entran en un marco: $4KB/4B = 2^{10}$
- ✓ Tamaño de tabla de páginas
 - ◆ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = $2^{52}/2^{10} = 2^{42}$
 - ◆ Tamaño tabla de páginas del proceso = $2^{42} * 4bytes = 2^{54}$

Más de 16.000GB por proceso!!!



Tabla de páginas - Tabla de 2 niveles

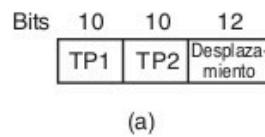
- El propósito de la tabla de páginas multinivel es dividir la tabla de páginas lineal en múltiples tablas de páginas
- Cada tabla de páginas suele tener el mismo tamaño pero se busca que tengan un menor número de páginas por tabla
- La idea general es que cada tabla sea más pequeña
- Se busca que la tabla de páginas no ocupe demasiada memoria RAM
- Además solo se carga una parcialidad de la tabla de páginas (solo lo que se necesite resolver)
- Existe un esquema de direccionamientos indirectos



Ejemplo: mapeo en memoria de tabla de páginas de 2 niveles

Se usan 3 páginas para referenciar

- 12MB de espacio
 - 4MB de datos
 - 4MB de texto
 - 4MB de stack



Las entradas sombreadas no se utilizan por no tener datos asignados

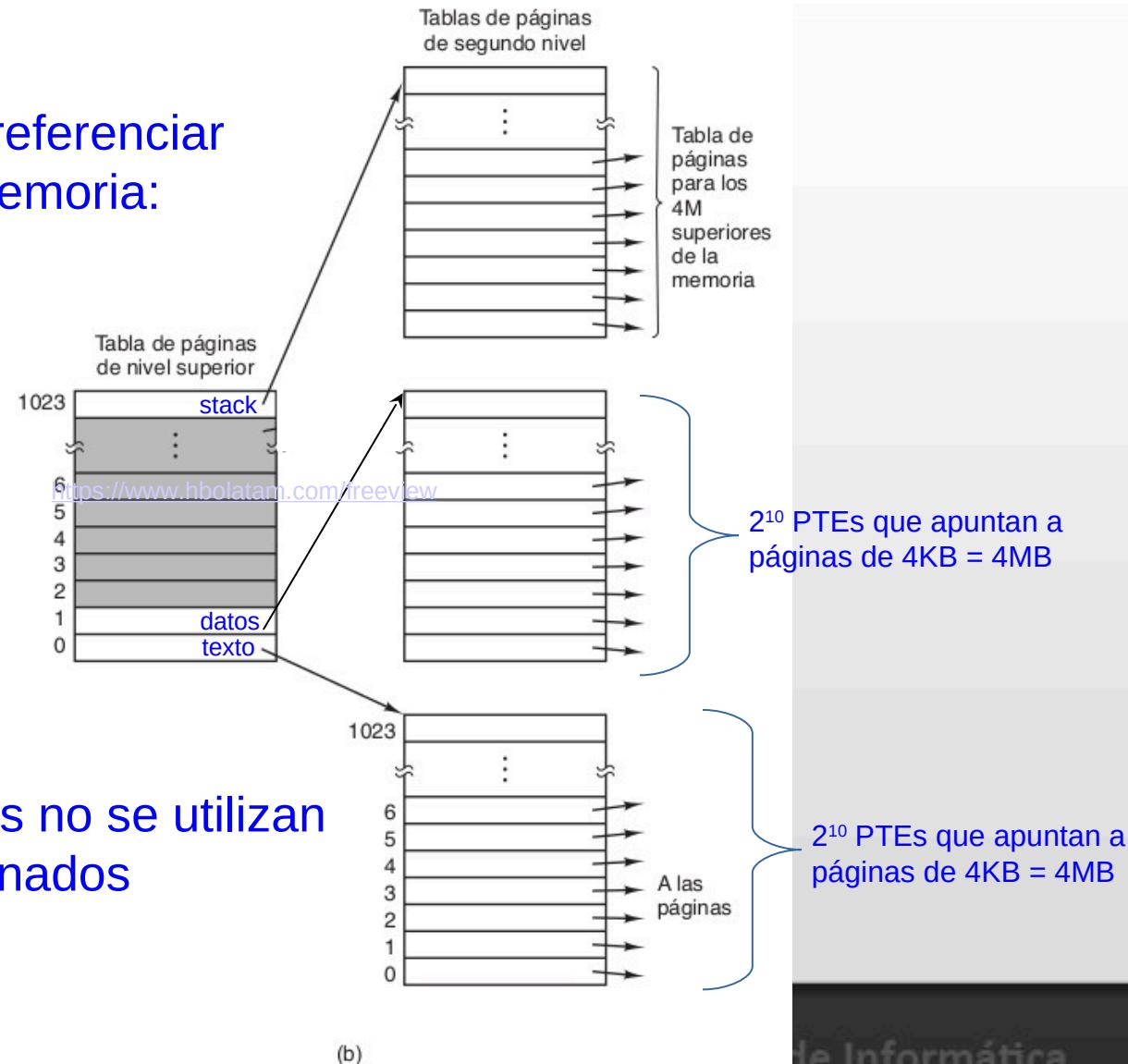
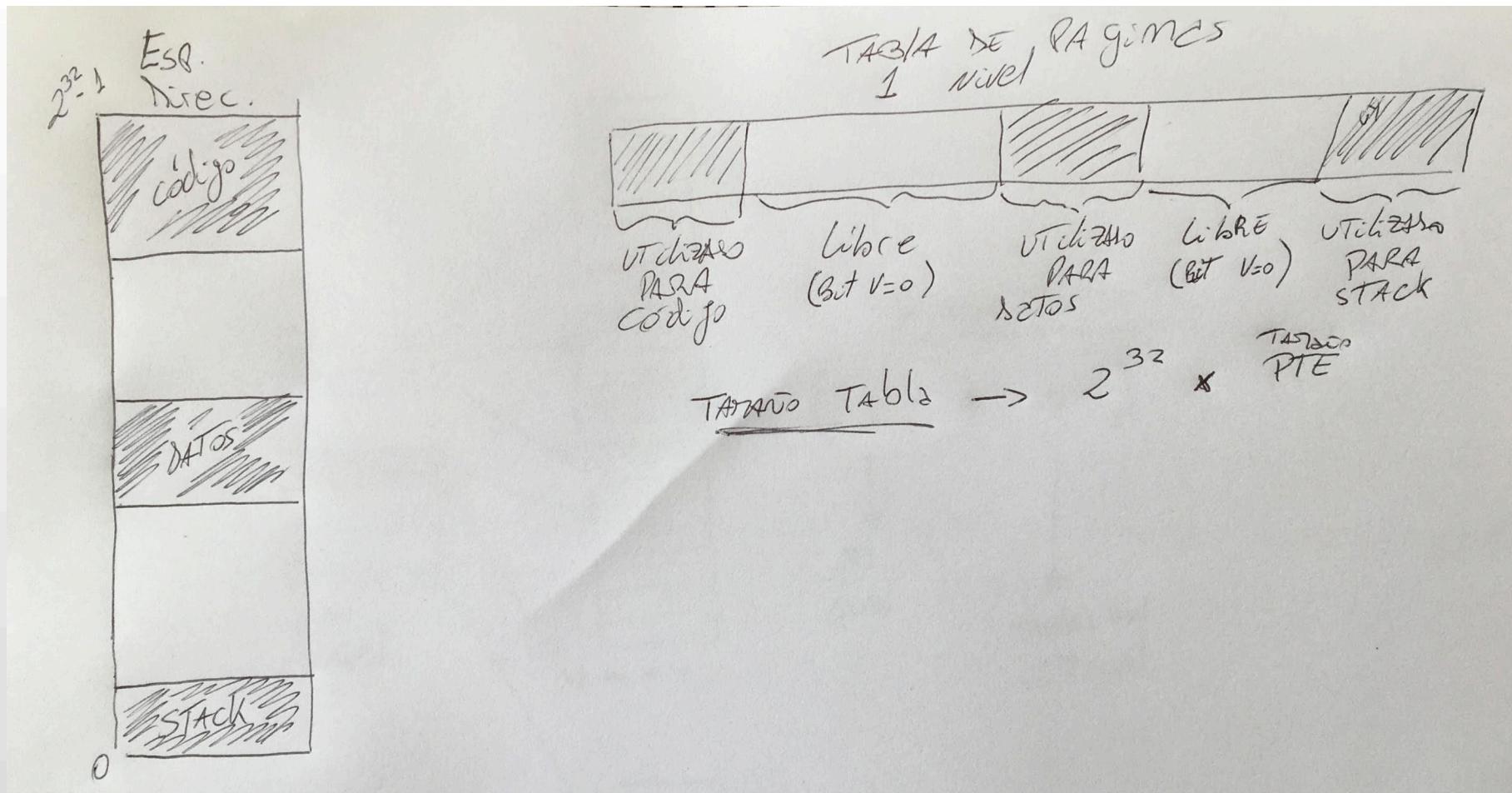


Figura 3-13. (a) Una dirección de 32 bits con dos campos de tablas de páginas. (b) Tablas de páginas de dos niveles.

1 nivel vs. 2 niveles (1)



1 nivel vs. 2 niveles (2)

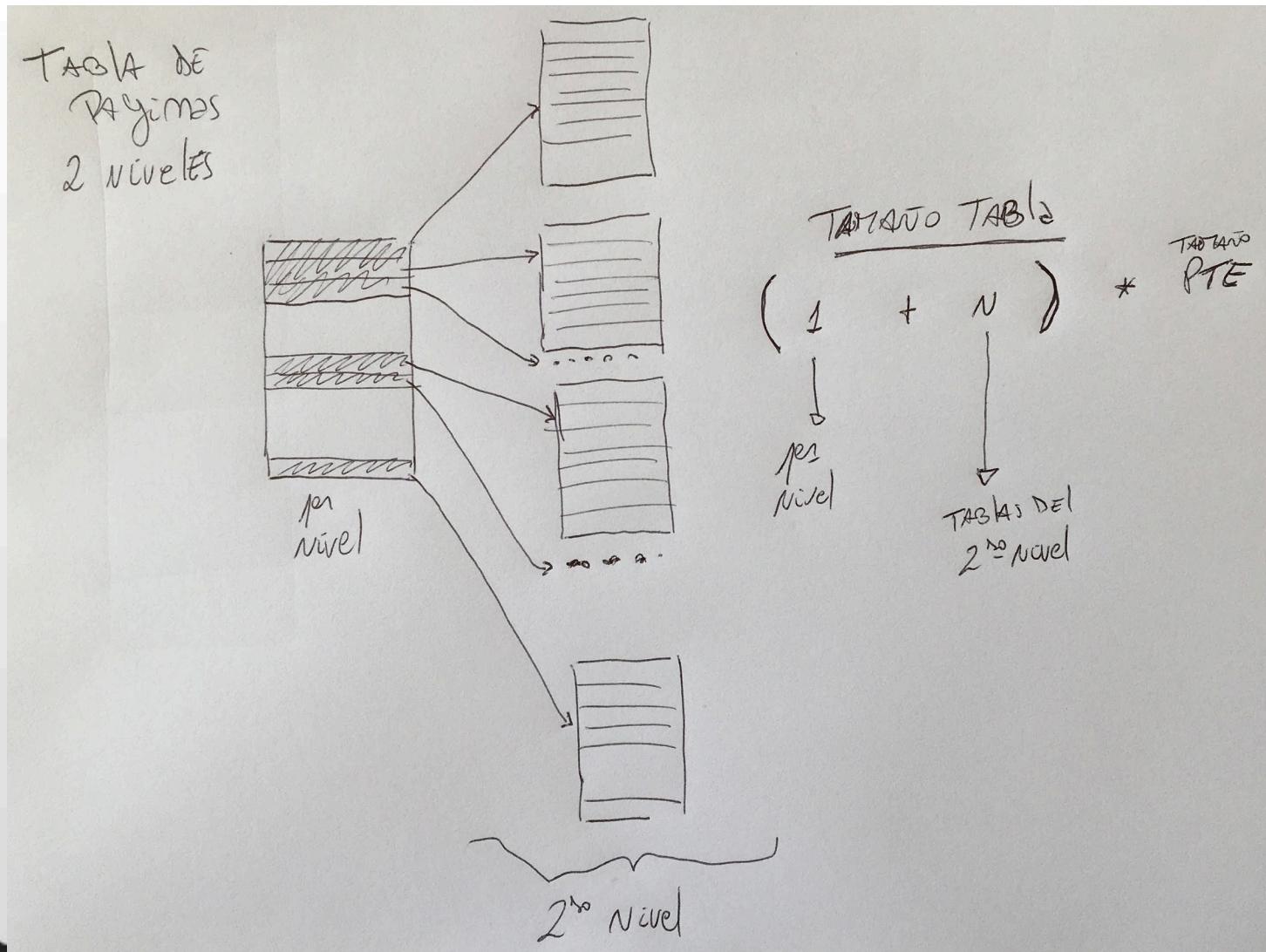
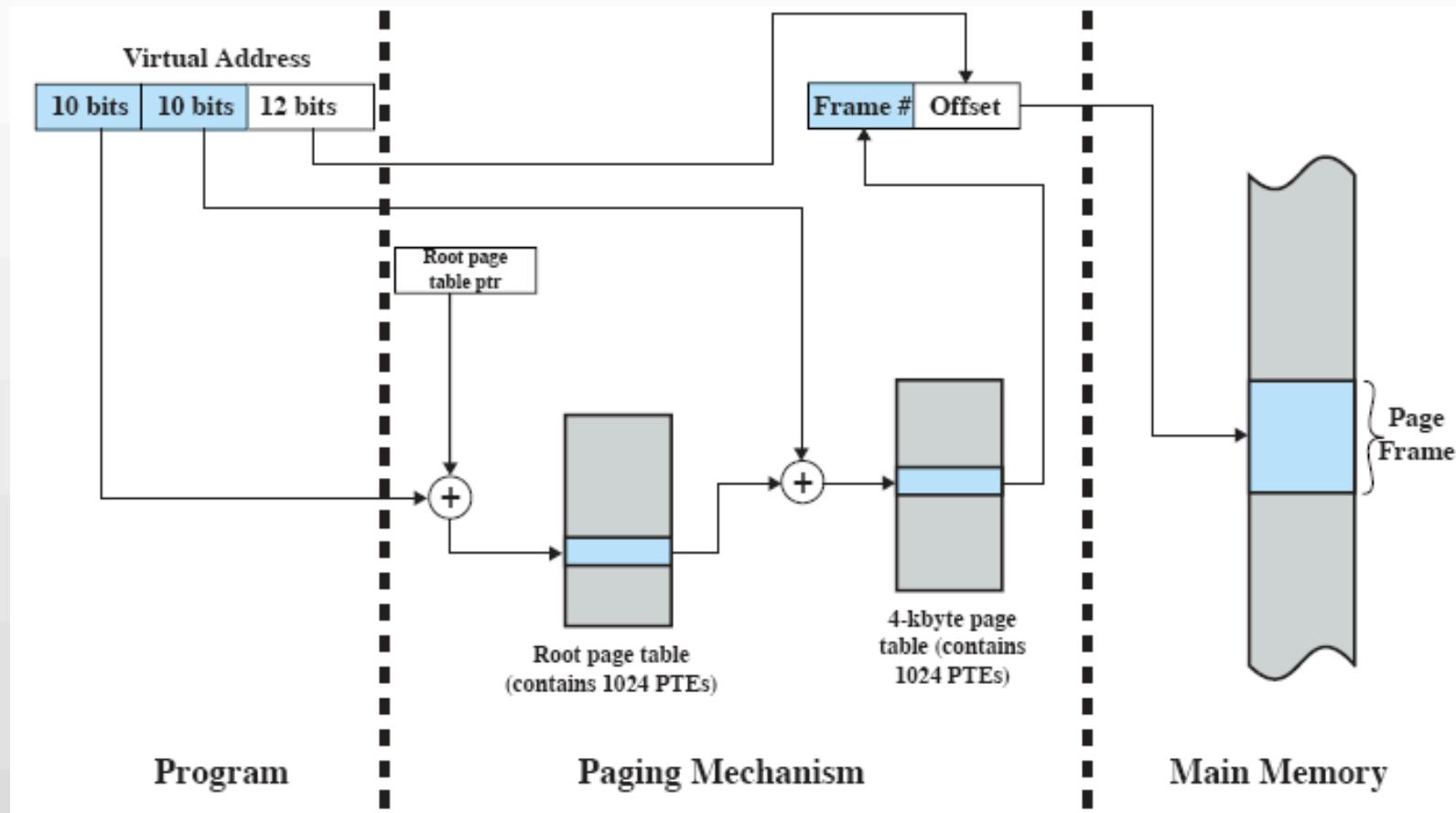
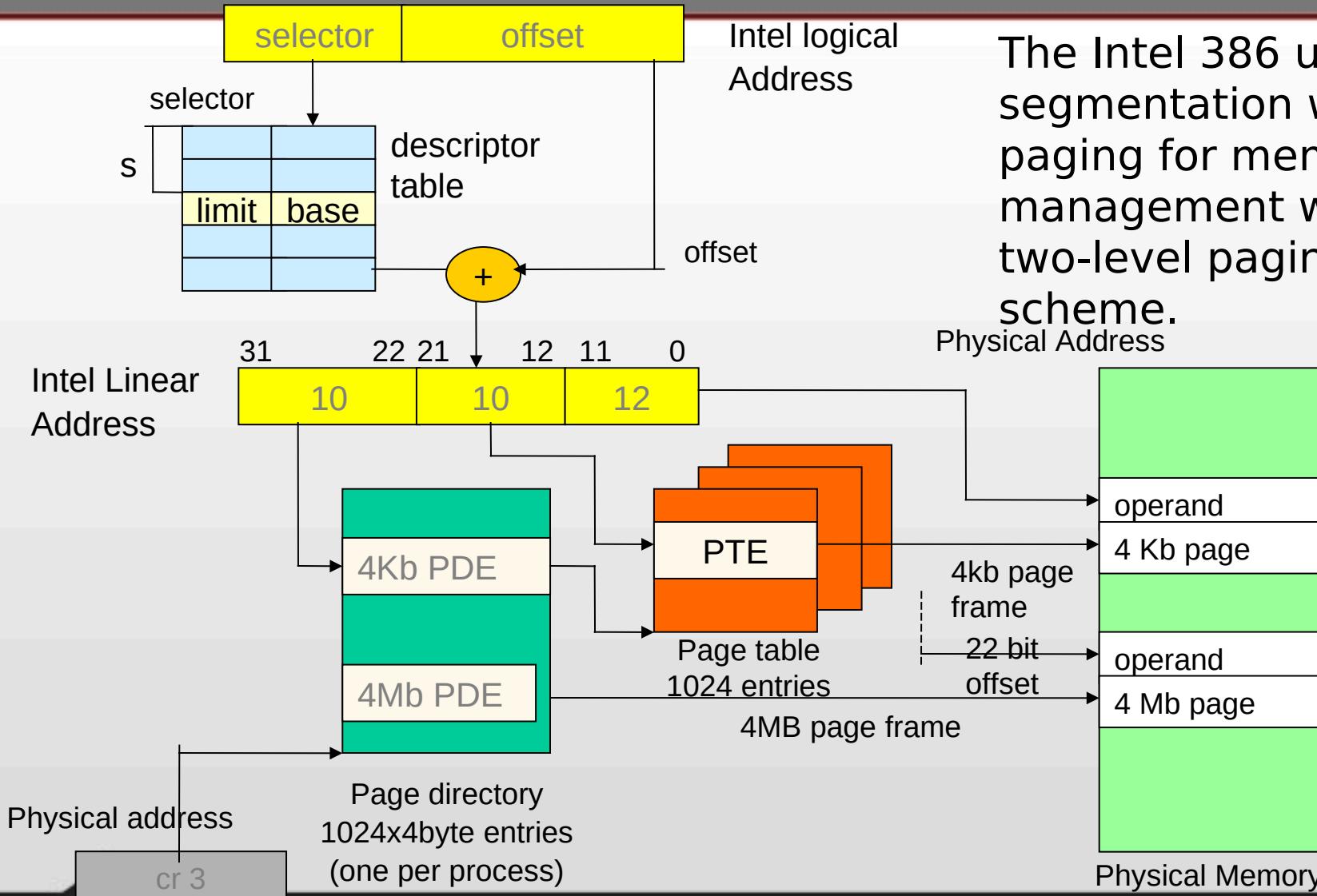


Tabla de Páginas - Tabla de 2 niveles



Intel 30386



Tablas de Páginas - x64

- ✓ Se usan 47 bits para direccionar
- ✓ Usa tabla de páginas de 4 niveles

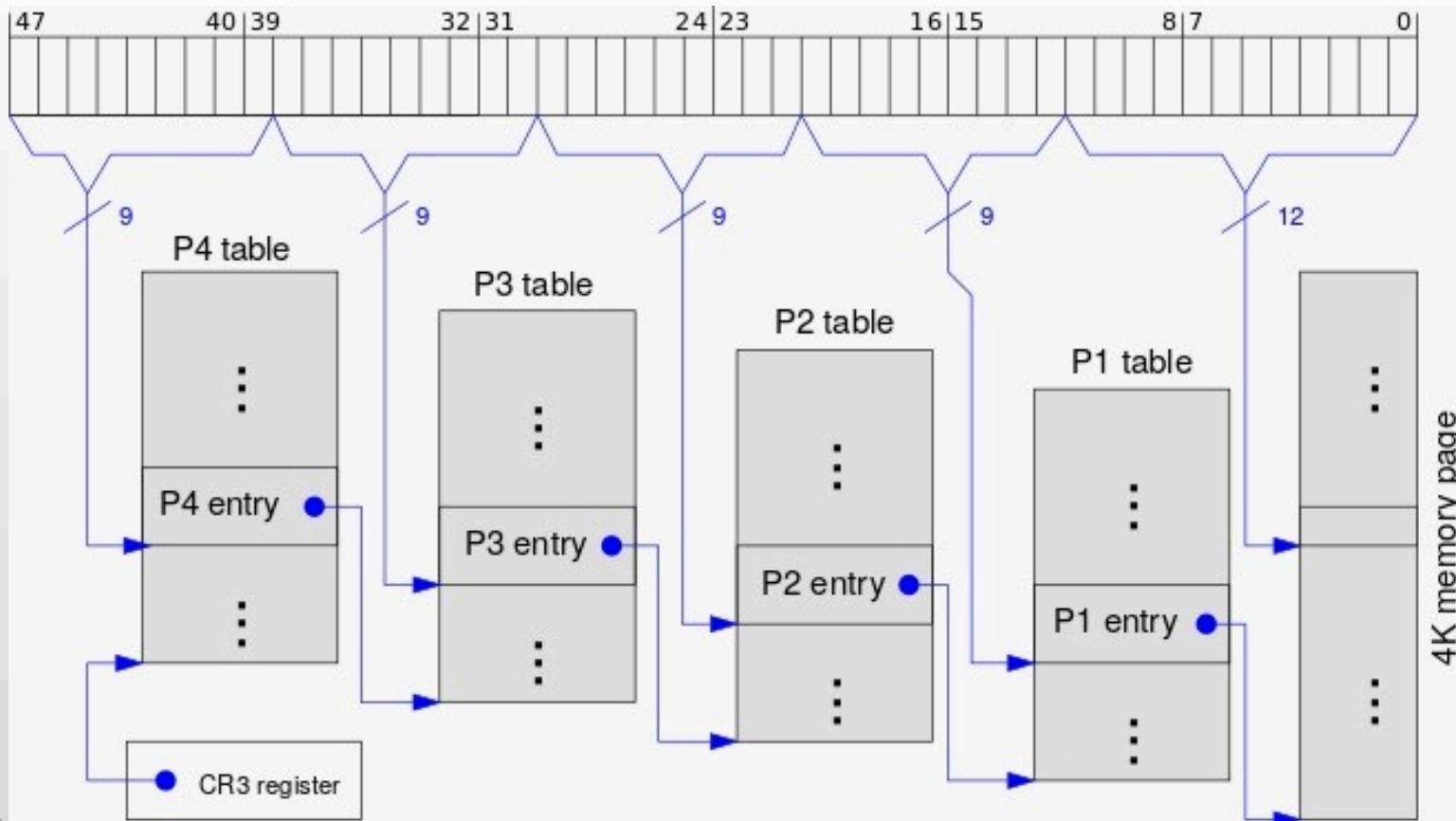


Tabla de Páginas (cont.) - Tabla invertida

- Utilizada en Arquitecturas donde el espacio de direcciones es muy grande
 - ✓ Las tablas de páginas ocuparían muchos niveles y la traducción sería costosa
 - ✓ Por esta razón se adopta esta técnica
- Por ejemplo, si el espacio de direcciones es de 2^{64} bytes, con páginas de 4 KB, necesitamos una tabla de páginas con 2^{52} entradas
- Si cada entrada es de 8 bytes, la tabla es de más de 30 millones de Gigabytes (30 PB)



Tabla de Paginas (cont.) - Tabla invertida

- Hay una entrada por cada marco de página en la memoria real. Es la visión inversa a la que veníamos viendo
- Hay una sola tabla para todo el sistema
- El espacio de direcciones de la tabla se refiera al espacio físico de la RAM, en vez del espacio de direcciones virtuales de un proceso
- Usada en PowerPC, UltraSPARC, y IA-64
- El número de página es transformado en un valor de HASH
- El HASH se usa como índice de la tabla invertida para encontrar el marco asociado
- Se define un mecanismo de encadenamiento para solucionar colisiones (cuando el hash da igual para 2 direcciones virtuales)



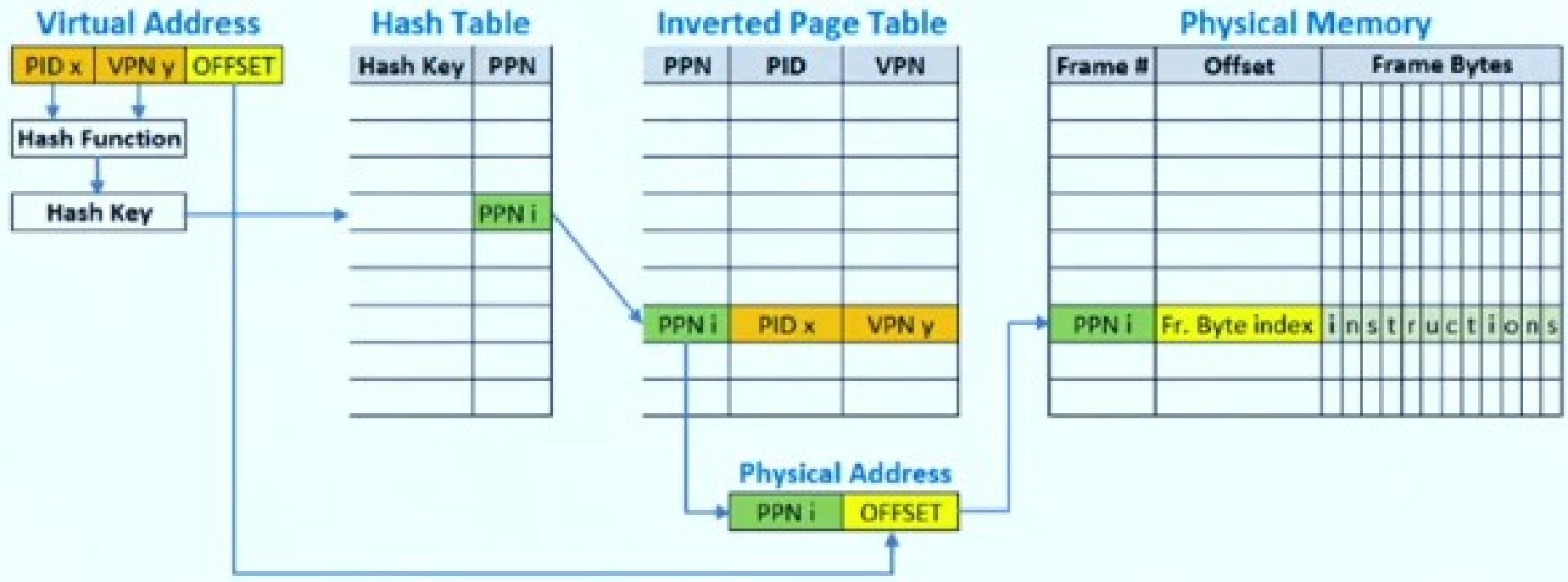
Tabla de Paginas (cont.) - Tabla invertida

Sólo se mantienen los PTEs de páginas presentes en memoria física

- ✓ La tabla invertida es organizada como tabla hash en memoria principal
 - ◆ Se busca indexadamente por número de página virtual
 - ◆ Si está presente en tabla, se extrae el marco de página y sus protecciones
 - ◆ Si no está presente en tabla, corresponde a un fallo de página



Tabla de Paginas (cont.) - Tabla invertida



<https://www.youtube.com/watch?v=2zEGiZga04g>



Tamaño de la Pagina

Pequeño

- ✓ Menor Fragmentación Interna.
- ✓ Más paginas requeridas por proceso → Tablas de páginas más grandes.
- ✓ Más paginas pueden residir en memoria

Grande

- ✓ Mayor Fragmentación interna
- ✓ La memoria secundaria está diseñada para transferir grandes bloques de datos más eficientemente → Mas rápido mover páginas hacia la memoria principal.



Tamaño de la Pagina (cont.)

Relación con la E/S

- ✓ Vel. De transferencia: 2 Mb/s
- ✓ Latencia: 8 ms
- ✓ Búsqueda: 20 ms

Página de 512 bytes

- 1 página → total: 28,2 ms
- Solo 0,2 ms de transferencia (1%)
- 2 páginas → 56,4 ms

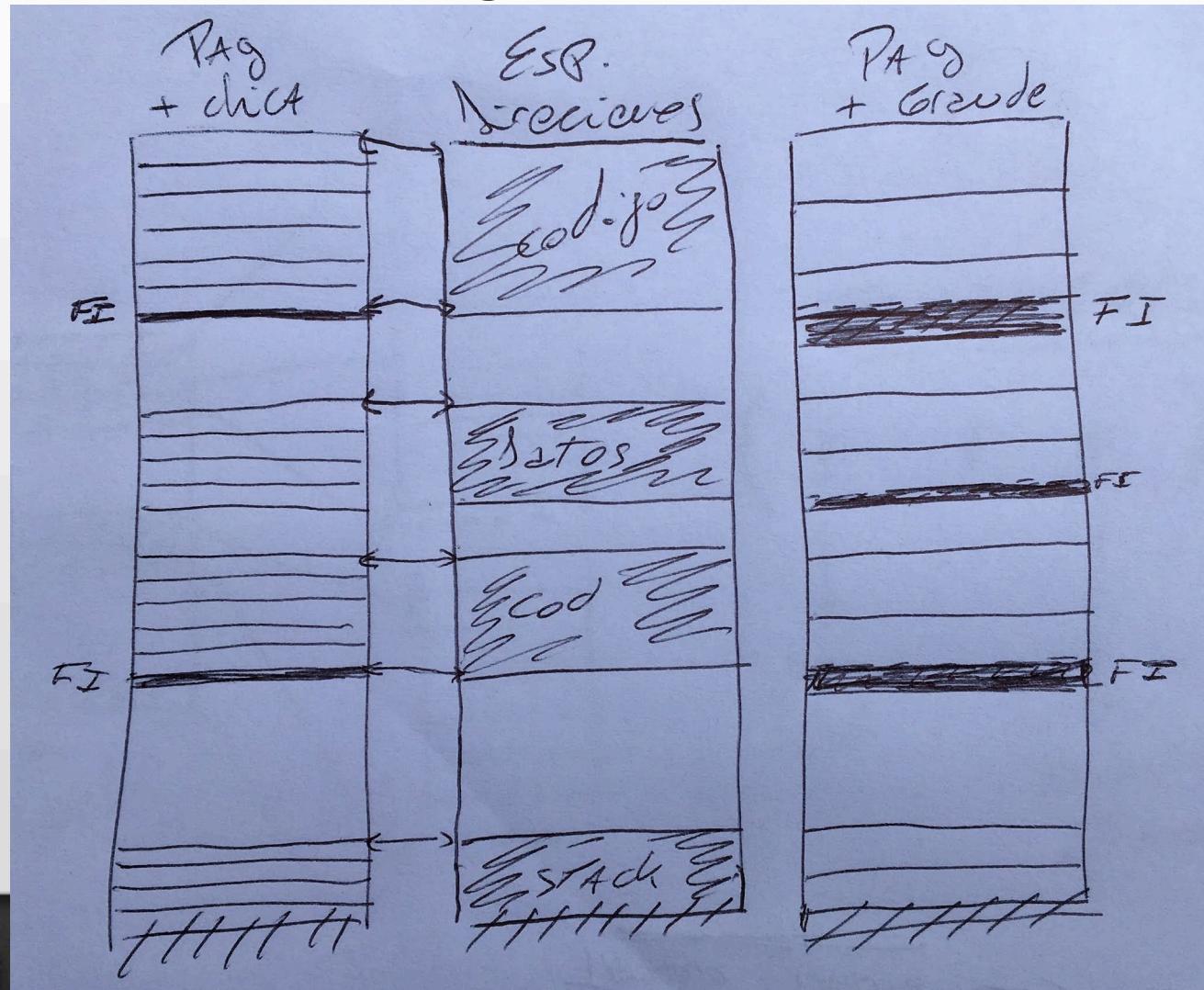
Página de 1024 bytes

- total: 28,4 ms
- Solo 0,4 ms de transferencia



Tamaño de la Pagina (cont.)

Relación con la fragmentación interna



Tamaño de la Pagina (cont)

Table 8.2 Example Page Sizes

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit word
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes



Translation Lookaside Buffer

- ✓ Cada referencia en el espacio virtual puede causar 2 (o más) accesos a la memoria física.
 - ✓ Uno (o más) para obtener la entrada en tabla de páginas
 - ✓ Uno para obtener los datos
- ✓ Para solucionar este problema, una memoria cache de alta velocidad es usada para almacenar entradas de páginas
 - ✓ TLB



Translation Lookaside Buffer (cont.)

- El buffer de traducción adelantada contiene las entradas de la tabla de páginas que fueron usadas más recientemente.
- Dada una dirección virtual, el procesador examina la TLB
- Si la entrada de la tabla de páginas se encuentra en la TLB (hit), es obtenido el frame y armada la dirección física

Translation Lookaside Buffer (cont.)

- Si la entrada no es encontrada en la TLB (miss), el número de página es usado como índice en la tabla de páginas del proceso.
- Se controla si la página está en la memoria
 - ✓ Si no está, se genera un Page Fault
- La TLB es actualizada para incluir la nueva entrada
- El cambio de contexto genera la invalidación de las entradas de la TLB
TLB ← Analizar.
¿Qué ocurre si el Quantum en Round Robin es chico? ¿y al contrario?



Translation Lookaside Buffer (cont.)

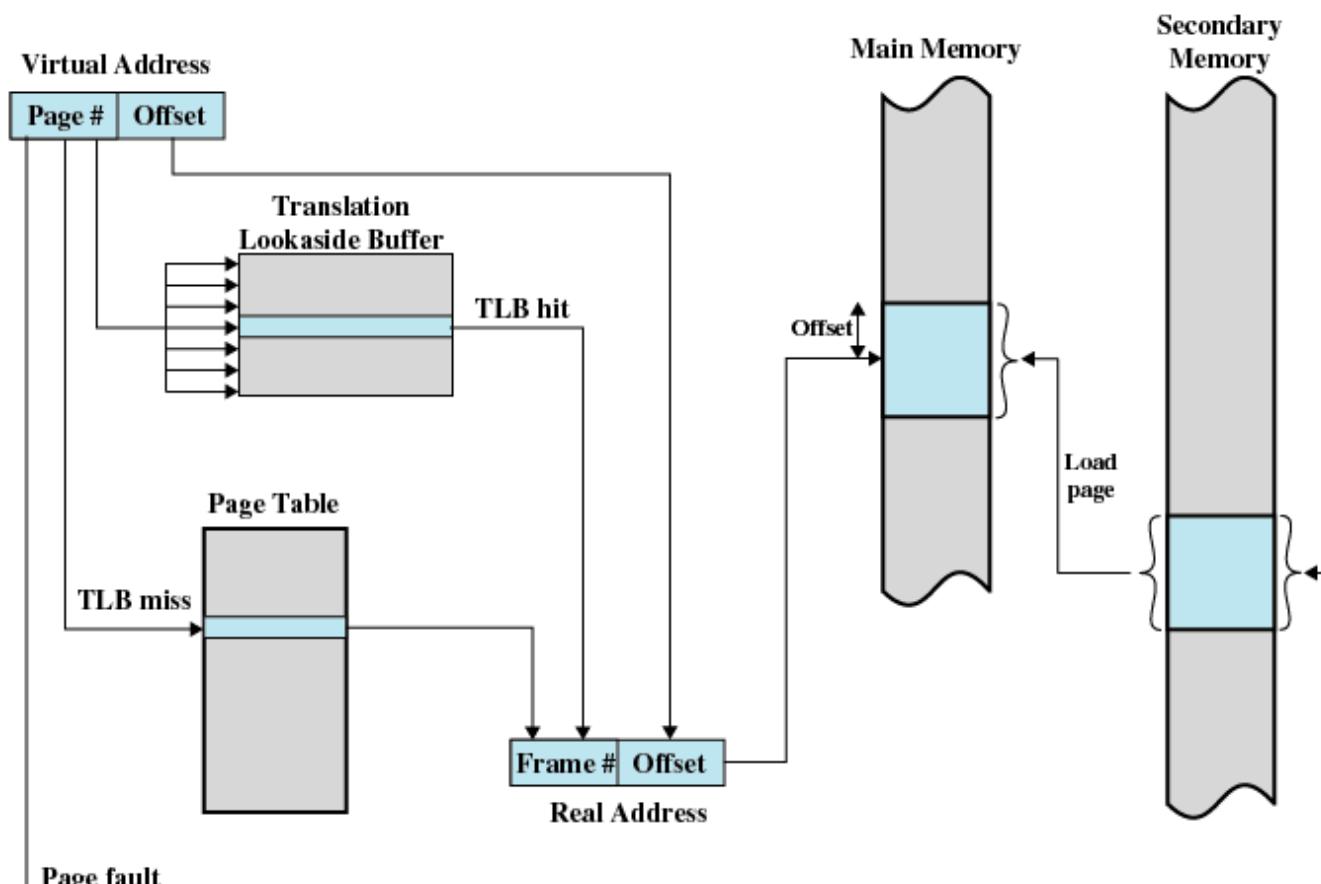


Figure 8.7 Use of a Translation Lookaside Buffer



Translation Lookaside Buffer (cont.)

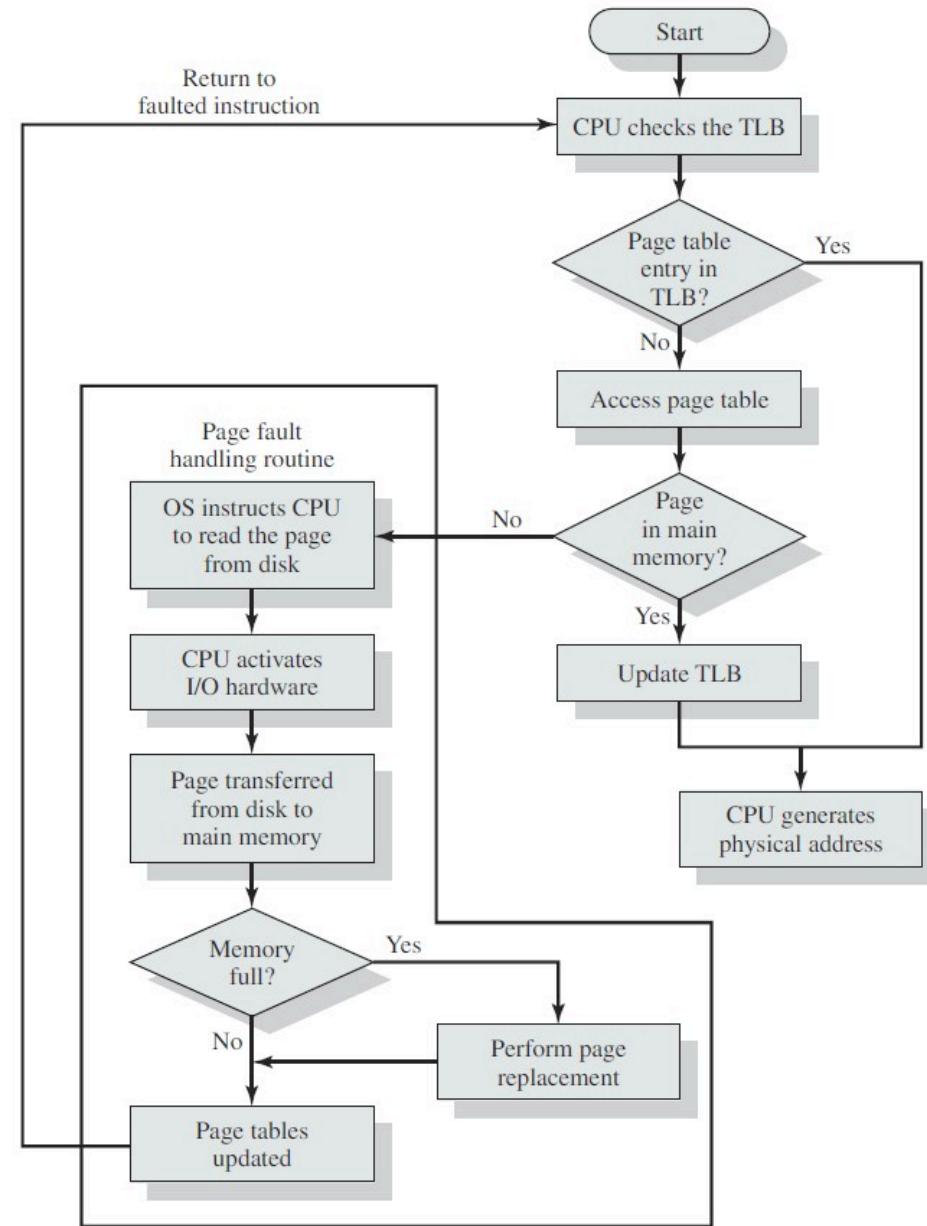


Figure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB)

Políticas en el manejo de MV

Table 8.4 Operating System Policies for Virtual Memory

Fetch Policy	<i>Cuando una página debe ser llevada a la memoria</i>	Resident Set Management
Demand paging		Resident set size
Prepaging		Fixed
	<i>Donde ubicarla (best-fit, first-fit, etc...)</i>	Variable
Placement Policy		<i>Cuántas páginas se traen a memoria</i>
Replacement Policy		Replacement Scope
Basic Algorithms		Global
Optimal	<i>Elección de víctima</i>	Local
Least recently used (LRU)		
First-in-first-out (FIFO)		
Clock		
Page Buffering		
		Cleaning Policy
		Demand
		Precleaning
		Load Control
		# de procesos en memoria
		Degree of multiprogramming



Asignación de Marcos

¿Cuántas páginas de un proceso se pueden encontrar en memoria?

- ✓ Tamaño del Conjunto Residente

Asignación Dinámica

- ✓ El número de marcos para cada proceso varía

Asignación Fija

- ✓ Número fijo de marcos para cada proceso



Asignación de Marcos - Asignación Fija

- Asignación equitativa – Ejemplo: si tengo 100 frames y 5 procesos, 20 frames para cada proceso
- Asignación Proporcional: Se asigna acorde al tamaño del proceso.

s_i = size of process p_i

$$S = \sum s_i$$

m = total number of frames

$$a_i = \text{allocation for } p_i = \frac{s_i}{S} \times m$$

$$m=64$$

$$s_1=10$$

$$s_2=127$$

$$a_1 = \frac{10}{137} \times 64 \approx 5$$

$$a_2 = \frac{127}{137} \times 64 \approx 59$$



Reemplazo de páginas

- Qué sucede si ocurre un fallo de página y todos los marcos están ocupados → “Se debe seleccionar una página víctima”
- ¿Cuál sería Reemplazo Optimo?
 - ✓ Que la página a ser removida no sea referenciada en un futuro próximo
- La mayoría de los reemplazos predicen el comportamiento futuro mirando el comportamiento pasado.



Alcance del Reemplazo

Reemplazo Global

- ✓ El fallo de página de un proceso puede reemplazar la página de cualquier proceso.
- ✓ El SO no controla la tasa de page-faults de cada proceso
- ✓ Puede tomar frames de otro proceso aumentando la cantidad de frames asignados a él.
- ✓ Un proceso de alta prioridad podría tomar los frames de un proceso de menor prioridad.



Alcance del Reemplazo (cont.)

Reemplazo Local

- ✓ El fallo de página de un proceso solo puede reemplazar sus propias páginas – De su Conjunto Residente
- ✓ No cambia la cantidad de frames asignados
- ✓ El SO puede determinar cuál es la tasa de page-faults de cada proceso
- ✓ Un proceso puede tener frames asignados que no usa, y no pueden ser usados por otros procesos.



Asignación y Alcance

Table 8.5 Resident Set Management

	Local Replacement	Global Replacement
Fixed Allocation	<ul style="list-style-type: none">• Number of frames allocated to a process is fixed.• Page to be replaced is chosen from among the frames allocated to that process.	<ul style="list-style-type: none">• Not possible.
Variable Allocation	<ul style="list-style-type: none">• The number of frames allocated to a process may be changed from time to time to maintain the working set of the process.• Page to be replaced is chosen from among the frames allocated to that process.	<ul style="list-style-type: none">• Page to be replaced is chosen from all available frames in main memory; this causes the size of the resident set of processes to vary.



Algoritmos de Reemplazo

- OPTIMO:** Es solo teórico
- FIFO:** Es el más sencillo
- LRU (Least Recently Used):** Requiere soporte del hardware para mantener timestamps de acceso a las páginas. Favorece a las páginas más recientemente accedidas
- 2da. Chance:** Un avance del FIFO tradicional que beneficia a las páginas más referenciadas
- NRU (Non Recently Used):**
 - ✓ Utiliza bits R y M
 - ✓ Favorece a las páginas que fueron usadas recientemente



Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de
Memoria - III



Versión: Octubre 2022

Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Paginación, Trashing, Working Set

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Thrashing (hiperpaginación)

- ✓ **Concepto:** decimos que un sistema está en *thrashing* cuando pasa más tiempo paginando que ejecutando procesos.
- ✓ Como consecuencia, hay una baja importante de performance en el sistema.

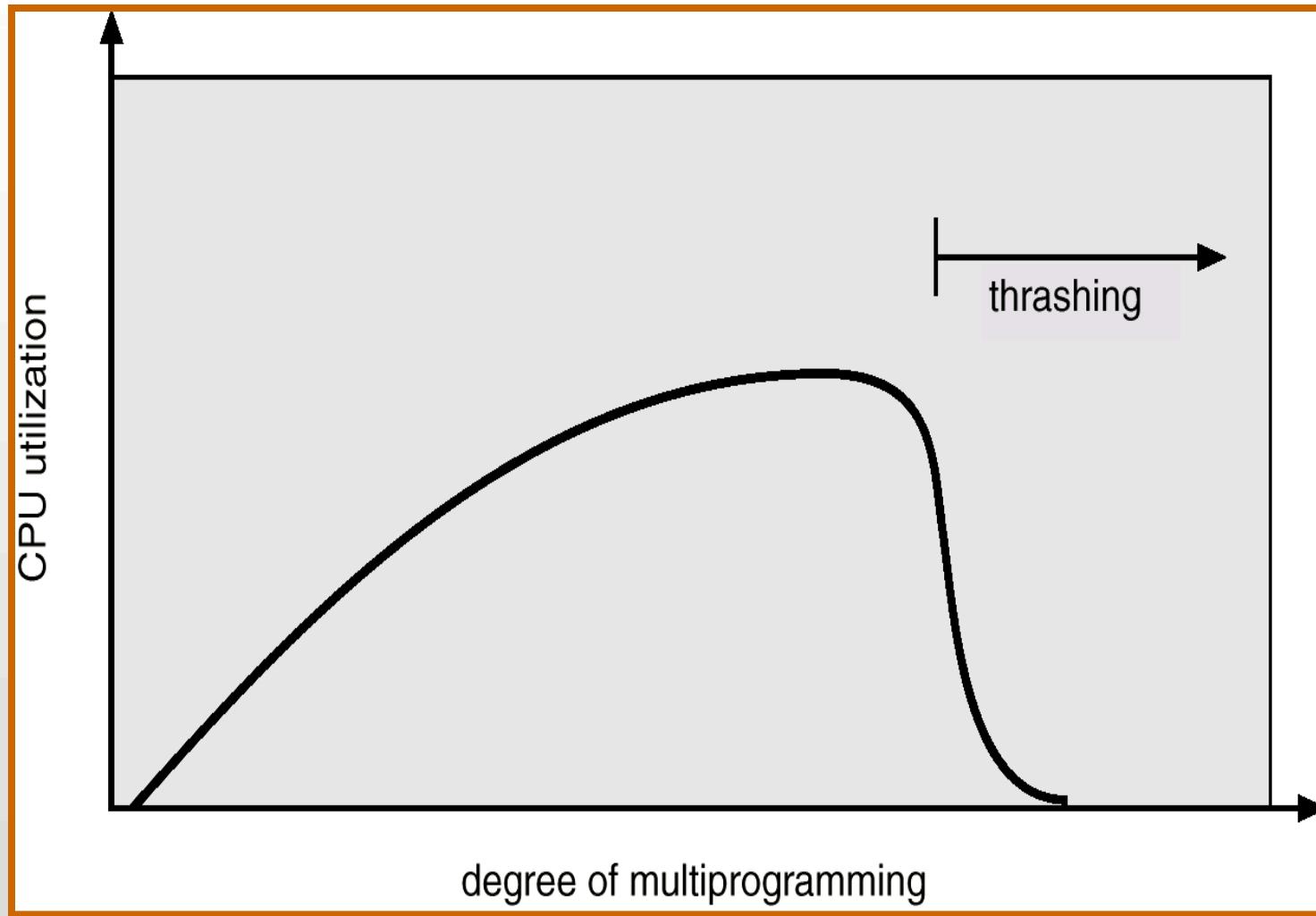


Ciclo del thrashing

- 1) El kernel monitorea el uso de la CPU.
- 2) Si hay baja utilización ⇒ aumenta el grado de multiprogramación.
- 3) Si el algoritmo de reemplazo es global, pueden sacarse frames a otros procesos.
- 4) Un proceso necesita más frames.
Comienzan los page-faults y robo de frames a otros procesos.
- 5) Por swapping de páginas, y encolamiento en dispositivos, baja el uso de la CPU.
- 6) Vuelve a 1).



Thrashing



El scheduler de CPU y el thrashing

- 1) Cuando se decrementa el uso de la CPU, el scheduler long term aumenta el grado de multiprogramación.
- 2) El nuevo proceso inicia nuevos page-faults, y por lo tanto, más actividad de paginado.
- 3) Se decrementa el uso de la CPU
- 4) Vuelve a 1).



Control del thrashing

- Se puede limitar el thrashing usando algoritmos de reemplazo local.
- Con este algoritmo, si un proceso entra en thrashing no roba frames a otros procesos.
- Si bien perjudica la performance del sistema, es controlable.



Conclusión sobre thrashing

- Si un proceso cuenta con todos los frames que necesita, no habría thrashing.
- Una manera de abordar esta problemática es utilizando la estrategia de **Working Set**, la cual se apoya en el modelo de localidad
- Otra estrategia con el mismo espíritu es la del algoritmo **PFF** (Frecuencia de Fallos de Página)



El modelo de localidad

- Cercanía de referencias o principio de cercanía
- Las referencias a datos y programa dentro de un proceso tienden a agruparse
- La localidad de un proceso en un momento dado se da por el conjunto de páginas que tiene en memoria en ese momento.
- En cortos períodos de tiempo, el proceso necesitará pocas “piezas” del proceso (por ejemplo, una página de instrucciones y otra de datos...)



El modelo de localidad

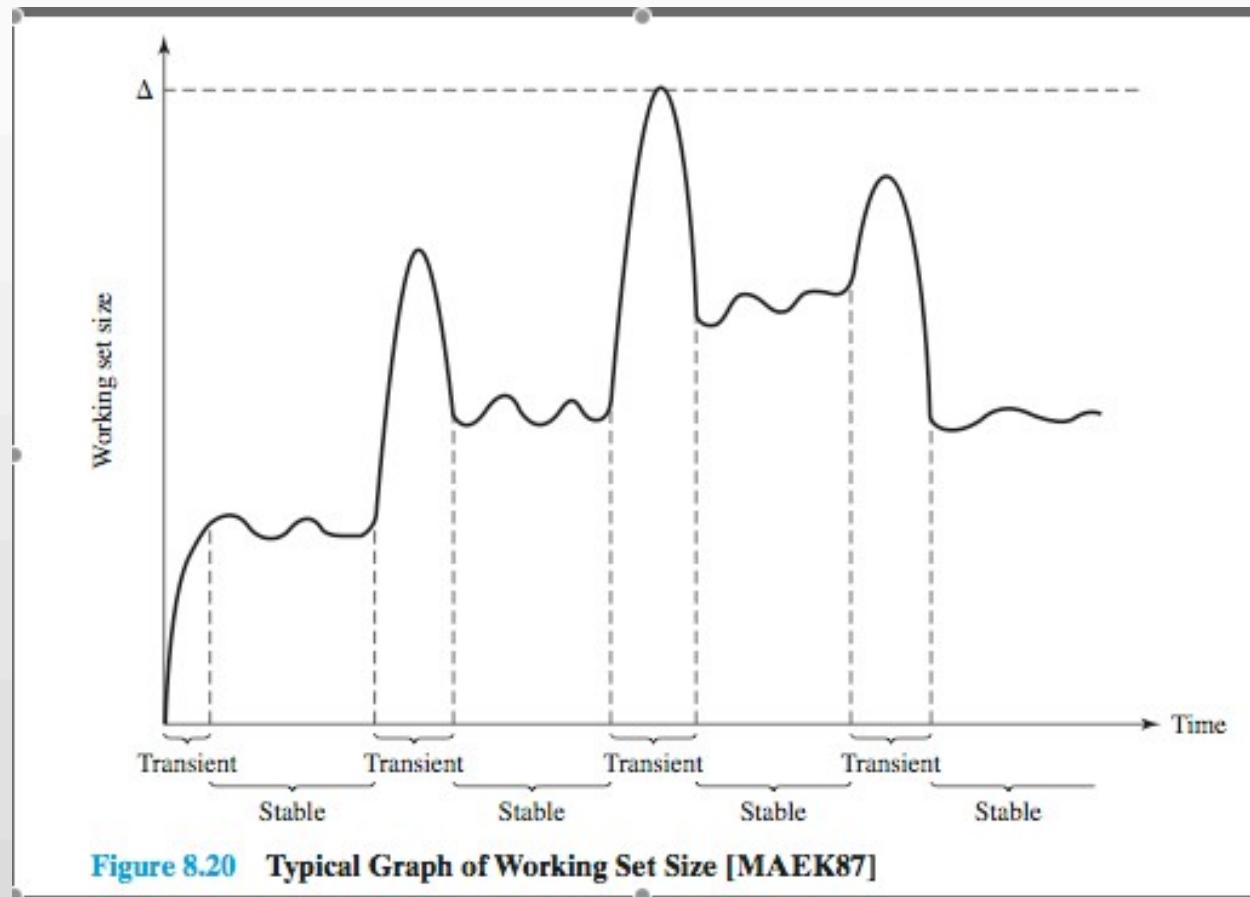


Figure 8.20 Typical Graph of Working Set Size [MAEK87]



El modelo de localidad (cont.)

- Un programa se compone de varias localidades.
- Ejemplo: Cada rutina será una nueva localidad: se mencionan sus direcciones (cerca) cuando se está ejecutando.
- Para prevenir la hiperactividad, un proceso debe tener en memoria sus páginas más activas (menos page faults).



El modelo de working set

- ✓ Se basa en el modelo de localidad.
- ✓ Ventana del working set (Δ): las referencias de memoria más recientes.
- ✓ Working set: es el conjunto de páginas que tienen las más recientes Δ referencias a páginas.

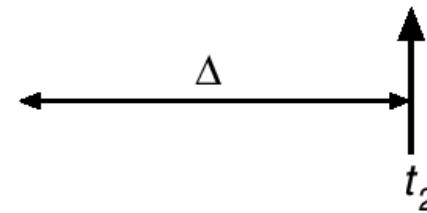
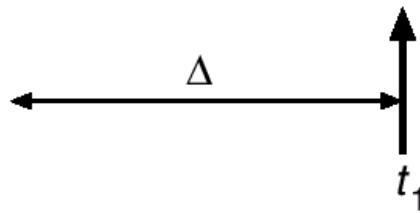


Ejemplo de working set

$\Delta=10$

page reference table

... 2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3 4 4 4 3 4 4 4 ...



La selección del Δ

- Δ chico: no cubrirá la localidad

- Δ grande: puede tomar varias localidades



Medida del working set

- $m = \text{cantidad frames disponibles}$
- $WSS_i = \text{tamaño del working set del proceso } p_i.$
- $\sum WSS_i = D;$
- $D = \text{demanda total de frames.}$
- Si $D > m$, habrá ***thrashing***.



Prevención del thrashing

- SO monitorea c/ proceso, dándole tantos frames hasta su WSS_i (medida del working set del proceso p_i)
- Si quedan frames, puede iniciar otro proceso.
- Si D crece, excediendo m, se elige un proceso para suspender, reasignándose sus frames...

Así, se mantiene alto el grado de multiprogramación optimizando el uso de la CPU.



Problema del modelo del WS

- Mantener un registro de los WSS_i
(medida del working set del proceso p_i)
- La ventana es móvil

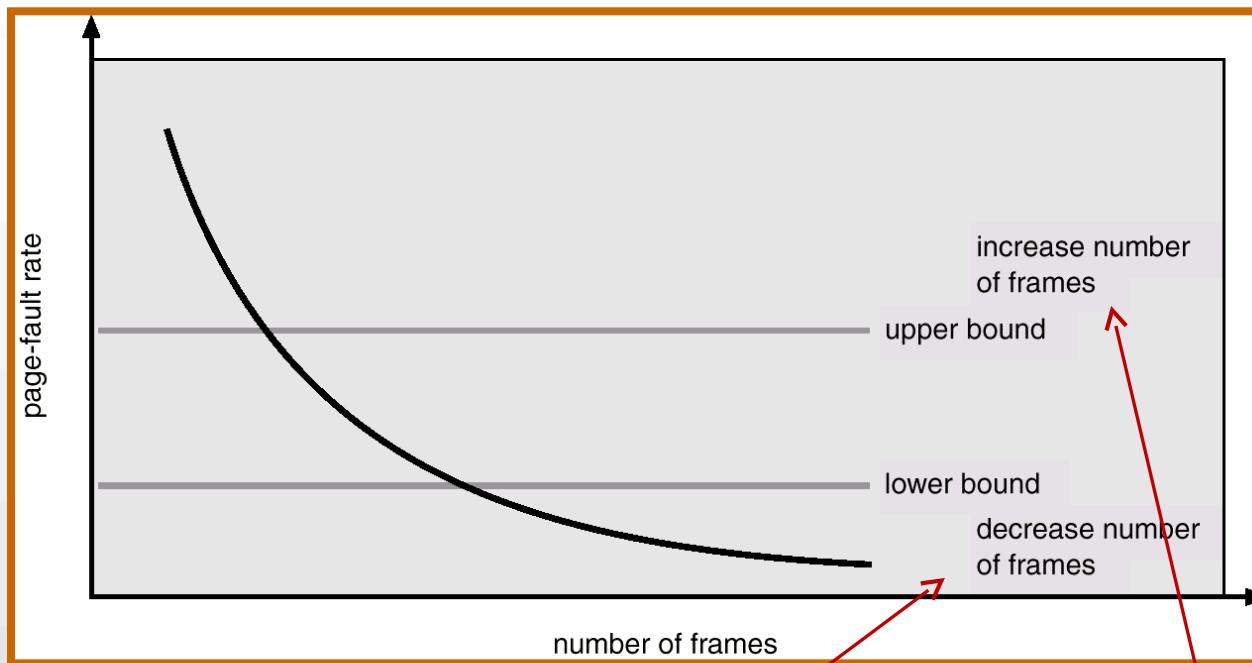


Prevención del thrashing por PFF

- ✓ La técnica PFF (Page Fault Frecuency o Frecuencia de Fallo de Páginas), en lugar de calcular el WS de los procesos, utiliza la tasa de fallos de página para estimar si el proceso tiene un conjunto residente que representa adecuadamente al WS.
- ✓ PFF: Frecuencia de page faults
- ✓ PFF alta ⇒ Se necesitan más frames
- ✓ PFF baja ⇒ Los procesos tienen frames asignados que le sobran



Esquema de PFF



- Establecer tasa de PF aceptable
 - ✓ Si la tasa actual es baja, el proceso puede perder frames
 - ✓ Si la tasa actual es alta, el proceso gana frames.



PFF (continuación)

- Establecer límites superior e inferior de las PFF's deseadas.
- Si se Excede PFF máx. ⇒ Se le asignan frames a mas al proceso, ya que el mismo genera muchos PF y probablemente los esté necesitando.
- Si la PFF está por debajo del mínimo ⇒ Se le pueden quitar frames al proceso para ser utilizados en los que necesitan mas.
- Se puede llegar a suspender un proceso si no hay más frames libres y todos los procesos estan por arriba de PFF Máx



Herramientas

Process Explorer - Sysinternals: www.sysinternals.com [ARBA\nicolas.delrio]

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
backgroundTaskHost...	Suspended	17.044 K	2.212 K	21732	Background Task Host	Microsoft Corporation
SearchHost.exe	Suspended	256.648 K	107.072 K	17364		Microsoft Corporation
backgroundTaskHost...	Suspended	17.024 K	2.204 K	32104	Background Task Host	Microsoft Corporation
OpenConsole.exe		2.656 K	12.784 K	28956		
WindowsTerminal.exe		29.432 K	66.820 K	3208		
RuntimeBroker.exe		3.152 K	11.520 K	19828	Runtime Broker	Microsoft Corporation
backgroundTaskHost...	Suspended	17.080 K	2.216 K	10668	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	16.932 K	2.212 K	13564	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.092 K	1.840 K	36972	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.136 K	1.840 K	43844	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.096 K	1.844 K	49216	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.024 K	1.840 K	49924	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.072 K	1.844 K	40796	Background Task Host	Microsoft Corporation
DataExchangeHost.exe		4.716 K	26.612 K	47844	Host de intercambio de datos	Microsoft Corporation
backgroundTaskHost...	Suspended	17.048 K	1.836 K	34248	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.076 K	1.824 K	38324	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	20.080 K	1.880 K	1584	Background Task Host	Microsoft Corporation
backgroundTaskHost...	Suspended	17.092 K	1.824 K	41996	Background Task Host	Microsoft Corporation
RuntimeBroker.exe		8.400 K	26.348 K	26832	Runtime Broker	Microsoft Corporation
backgroundTaskHost...	Suspended	3.396 K	3.776 K	23034	Background Task Host	Microsoft Corporation
smartscreen.exe		2.528 K	10.776 K	19680	SmartScreen de Windows D...	Microsoft Corporation
WmiPrvSE.exe		12.492 K	18.876 K	20528		
ScreenClippingHost.exe	0.12	7.104 K	32.884 K	27944		Microsoft Corporation
svchost.exe	< 0.01	39.376 K	31.412 K	1488	Proceso host para los servicios...	Microsoft Corporation
svchost.exe	< 0.01	5.928 K	10.164 K	1528	Proceso host para los servicios...	Microsoft Corporation
svchost.exe	< 0.01	29.384 K	16.840 K	1764	Proceso host para los servicios...	Microsoft Corporation
svchost.exe		2.240 K	4.304 K	1784	Proceso host para los servicios...	Microsoft Corporation
svchost.exe	< 0.01	3.280 K	4.988 K	1792	Proceso host para los servicios...	Microsoft Corporation
svchost.exe		12.020 K	7.152 K	1800	Proceso host para los servicios...	Microsoft Corporation

Permiten manipular:

- Working Set
- Otros parámetros

En GNU/Linux:

- Cat /proc/<PID>/statm
- Cat /proc/<PID>/status



Permiten analizar información de los procesos:

- Working Set, Delta, Fallos de página, espacio virtual vs espacio físico ocupado en memoria, entre otros parámetros

VMMap - www.sysinternals.com

File Edit View Tools Options Help

Process: chrome.exe
PID: 16724

Committed: 1.272.920 K

Private Bytes: 465.080 K

Working Set: 12.516 K

Type	Size	Committed	Private	Total WS	Private WS	Shareable WS	Unshareable WS
Total	2.253.645.504 K	1.272.920 K	465.080 K	12.516 K	6.308 K	6.208 K	2 K
Free	135.185.307.904 K						
Heap	38.136 K	20.032 K	19.968 K	1.012 K	1.008 K		
Image	423.528 K	423.528 K	10.940 K	5.992 K	372 K	5.62 K	
Managed Heap							
Mapped File	168.604 K	168.604 K		28 K			
Page Table							
Private Data	104.913.684 K	432.748 K	432.460 K	4.876 K	4.868 K		
Shareable	2.147.650.380 K	226.296 K		548 K			54 K
Stack	434.176 K	1.712 K	1.712 K	60 K	60 K		
Unusable	16.996 K						

Address	Type	Size	Committed	Private	Total WS	Private WS	Shareable WS	Unshareable WS
0000000000000000	Free	1.259.9...						
0000000004CE70000	Private Data	148 K	148 K	148 K				
0000000004CE95000	Unusable	44 K						
0000000004CEA0000	Free	522.68...						
0000000006CD10000	Image (ASLR)	228 K	228 K	4 K	40 K	16 K	24 K	24 K
0000000006CD49000	Unusable	28 K						
0000000006CD50000	Free	313.92...						

PFF y estructura de un programa

✓ Ejemplo:

- ✓ **int A[][] = new int[1024][1024];**
- ✓ Cada fila se almacena en una página
- ✓ Programa 1:

```
for (j = 0; j < A.length; j++)
    for (i = 0; i < A.length; i++)
        A[i, j] = 0;
```

1024 x 1024 page faults

} Recorrido por filas

- ✓ Programa 2:

```
for (i = 0; i < A.length; i++)
    for (j = 0; j < A.length; j++)
        A[i, j] = 0;
```

} Recorrido por columnas

1024 page faults



Demonio de Paginación

- Proceso creado por el SO durante el arranque que apoya a la administración de la memoria
- Se ejecuta cuando el sistema tiene una baja utilización o algún parámetro de la memoria lo indica
 - ✓ Poca memoria libre
 - ✓ Mucha memoria modificada
- Tareas:
 - ✓ Limpiar páginas modificadas sincronizándolas con el swap
 - ✓ Reducir el tiempo de swap posterior ya que las páginas están “limpias”
 - ✓ Reducir el tiempo de transferencia al sincronizar varias páginas contiguas.
 - ✓ Mantener un cierto número de marcos libres en el sistema.
 - ✓ Demorar la liberación de una página hasta que haga falta realmente



Demonio de Paginación - Ejemplos

En Linux

- ✓ Proceso “**kswapd**”

En Windows

- ✓ Proceso “**system**”

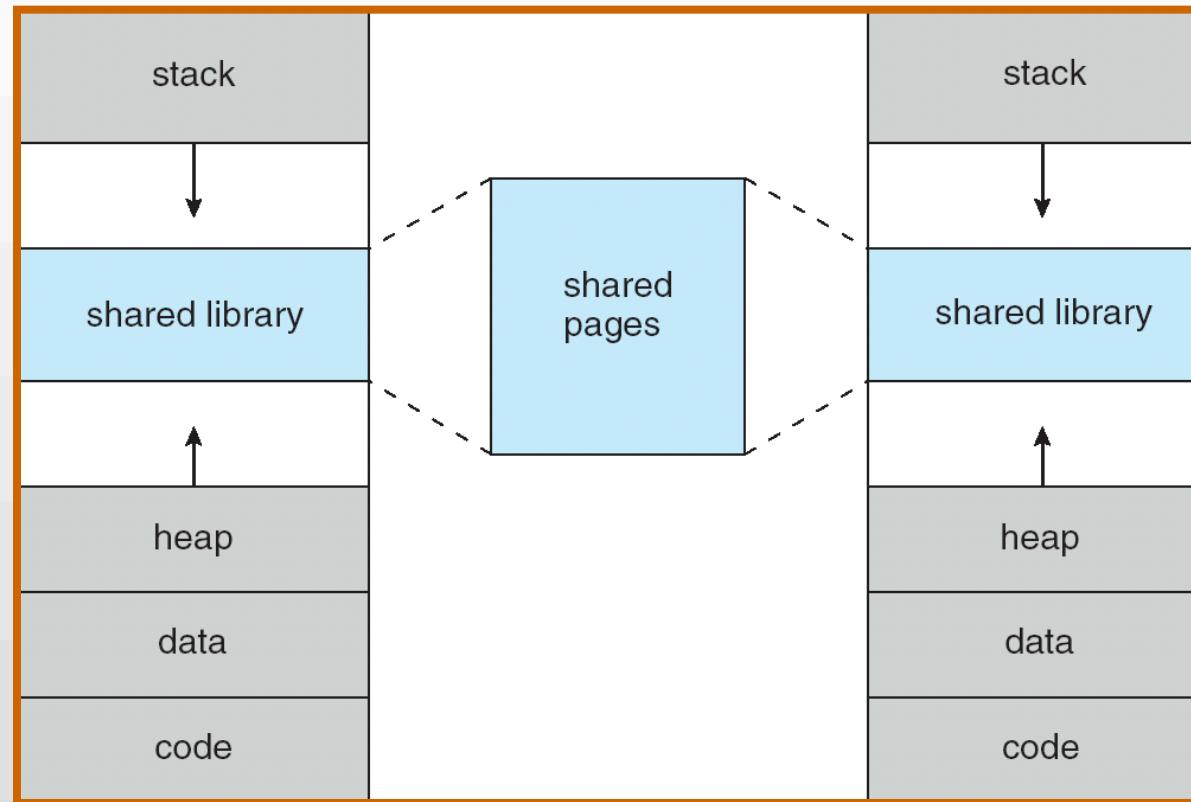


Memoria Compartida

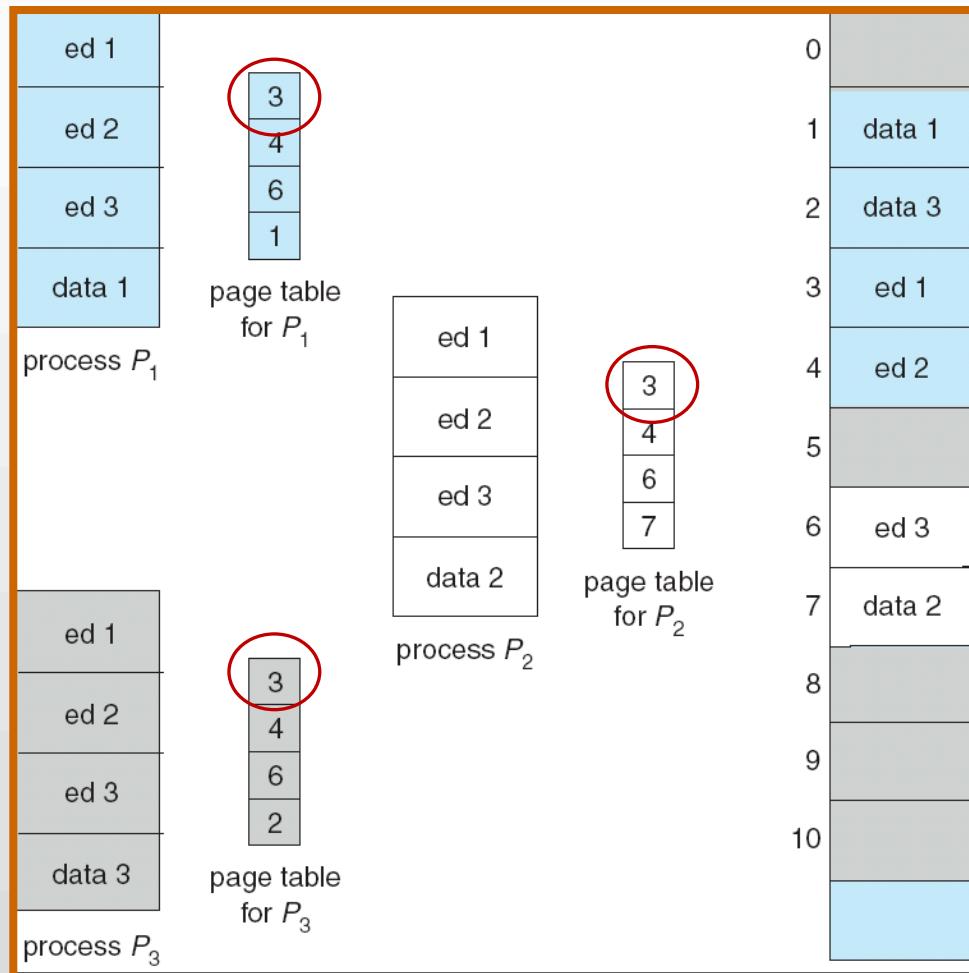
- ✓ Gracias al uso de la tabla de páginas varios procesos pueden compartir un marco de memoria; para ello ese marco debe estar asociado a una página en la tabla de páginas de cada proceso
 - ✓ El número de página asociado al marco puede ser diferente en cada proceso
- ✓ Código compartido**
- ✓ Los procesos comparten una copia de código (sólo lectura) por ej. editores de texto, compiladores, etc
 - ✓ Los datos son privados a cada proceso y se encuentran en páginas no compartidas



Memoria Compartida (cont.)



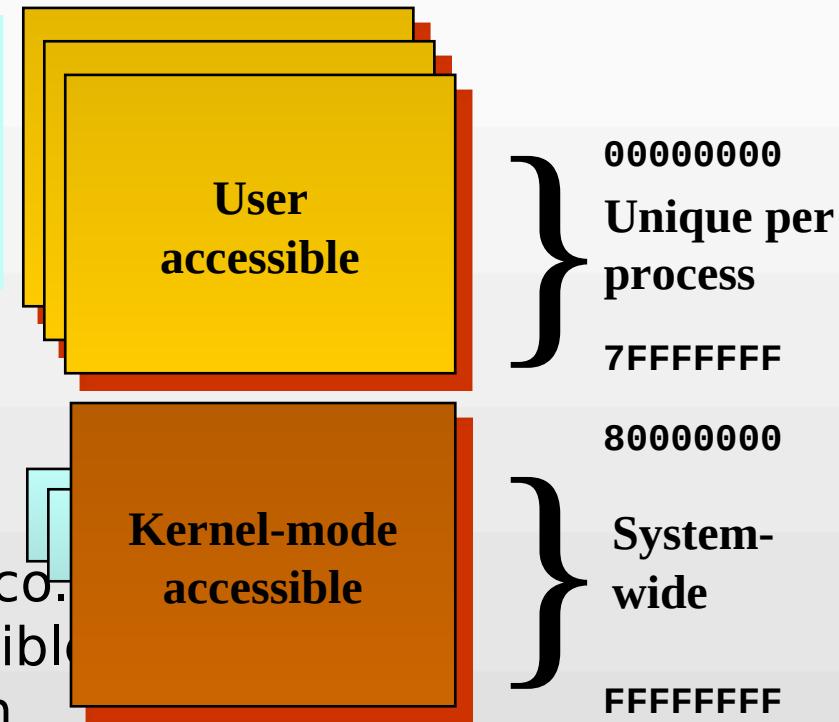
Memoria Compartida (cont.)



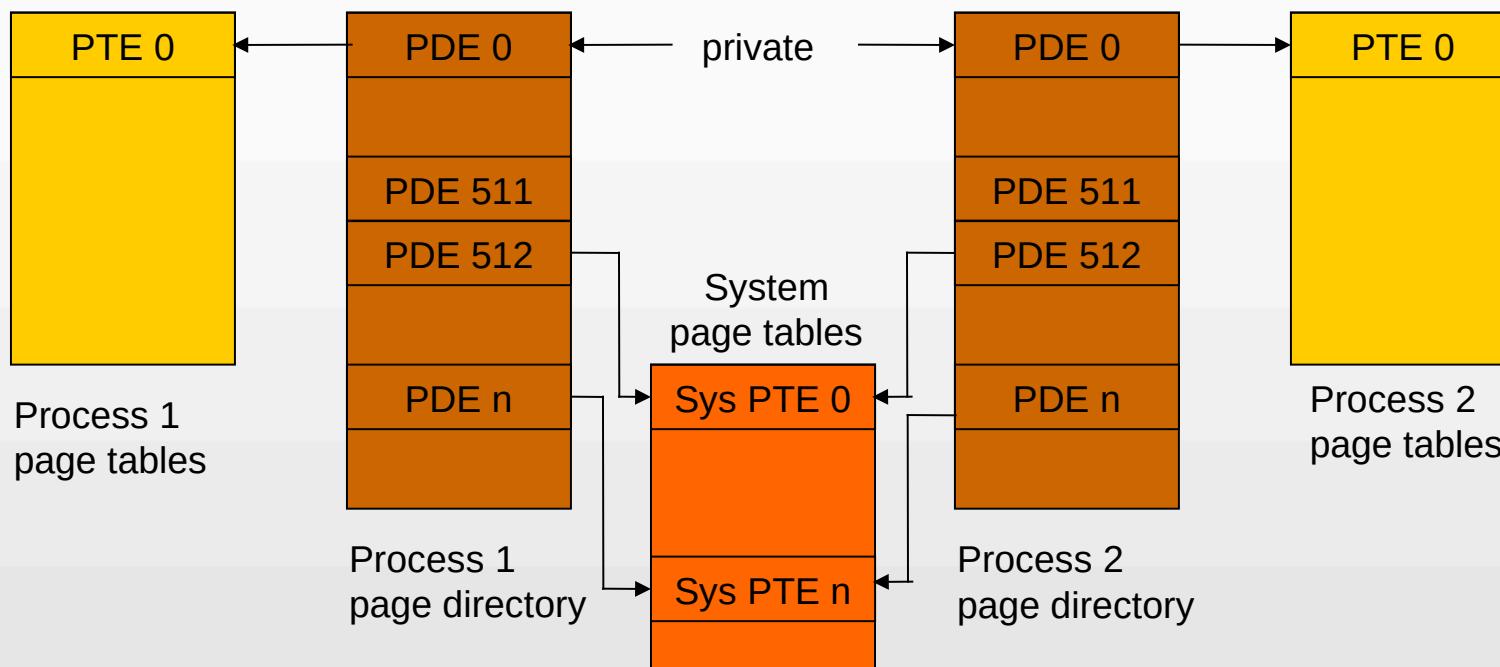
Memoria Compartida - Ej. Windows

Process space contains (32 Bits):

- ✓ The application you're running (.EXE and .DLLs)
- ✓ All static storage defined by the application
- ✓ Divide el espacio de direcciones lógico. Los 2 GB de nivel superior son accesibles en modo kernel, los 2 GB inferiores en modo usuario:
 - ✓ Los 2 GB superiores son accesibles por todos los procesos, se comparten



Memoria Compartida - Ej. Windows (cont.)



- ✓ On process creation, system space page directory entries point to existing system page tables



Mapeo de Archivo en Memoria

- ✓ Técnica que permite a un proceso asociar el contenido de un archivo a una región de su espacio de direcciones virtuales
- ✓ El contenido del archivo no se sube a memoria hasta que se generan Page Faults
- ✓ El contenido de la pagina que genera el PF es obtenido desde el archivo asociado
 - ✓ No del área de intercambio



Mapeo de Archivo en Memoria (cont.)

- Cuando el proceso termina o el archivo se libera, las páginas modificadas son escritas en el archivo correspondiente
- Permite realizar E/S de una manera alternativa a usar operaciones directamente sobre el Sistema de Archivos
- Es utilizado comúnmente para asociar librerías compartidas o DLLs



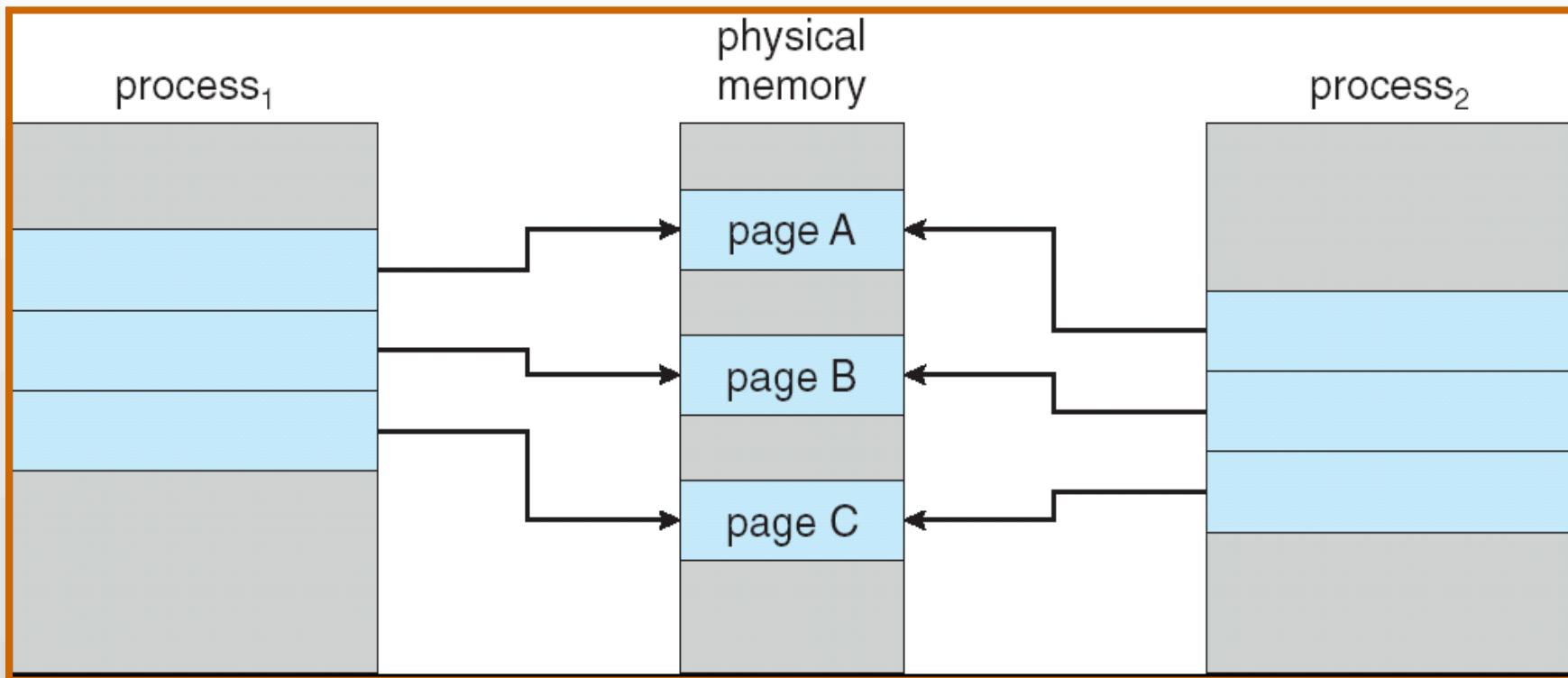
Copia en Escritura

- La copia en escritura (Copy-on-Write, COW) permite a los procesos compartir inicialmente las mismas páginas de memoria
 - ✓ Si uno de ellos modifica una página compartida la página es copiada
 - ✓ En fork, permite inicialmente que padre e hijo utilicen las mismas páginas sin necesidad de duplicación.
- COW permite crear procesos de forma más eficiente debido a que sólo las páginas modificadas son duplicadas



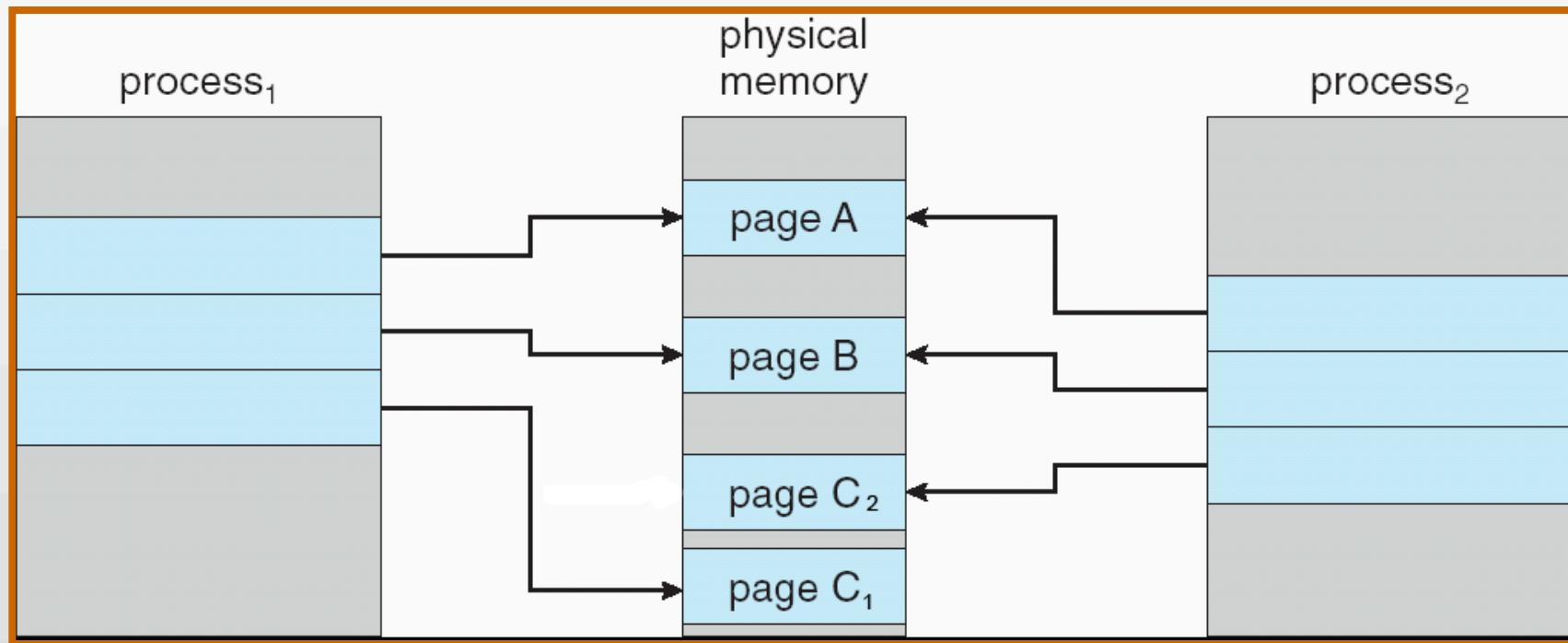
Copia en Escritura (cont.)

El Proceso 1 Modifica la Página C (Antes)



Copia en Escritura (cont.)

El Proceso 1 Modifica la Página C (Después)



Área de Intercambio

Sobre el Área utilizada

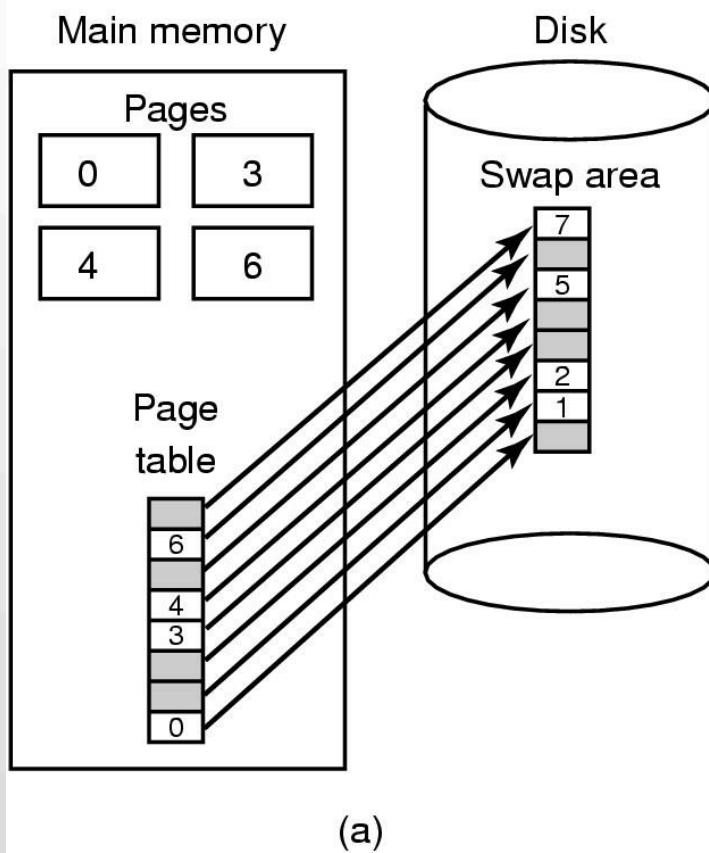
- ✓ Área dedicada, separada del Sistema de Archivos (Por ejemplo, en Linux)
- ✓ Un archivo dentro del Sistema de Archivos (Por ejemplo, Windows)

Técnicas para la Administración:

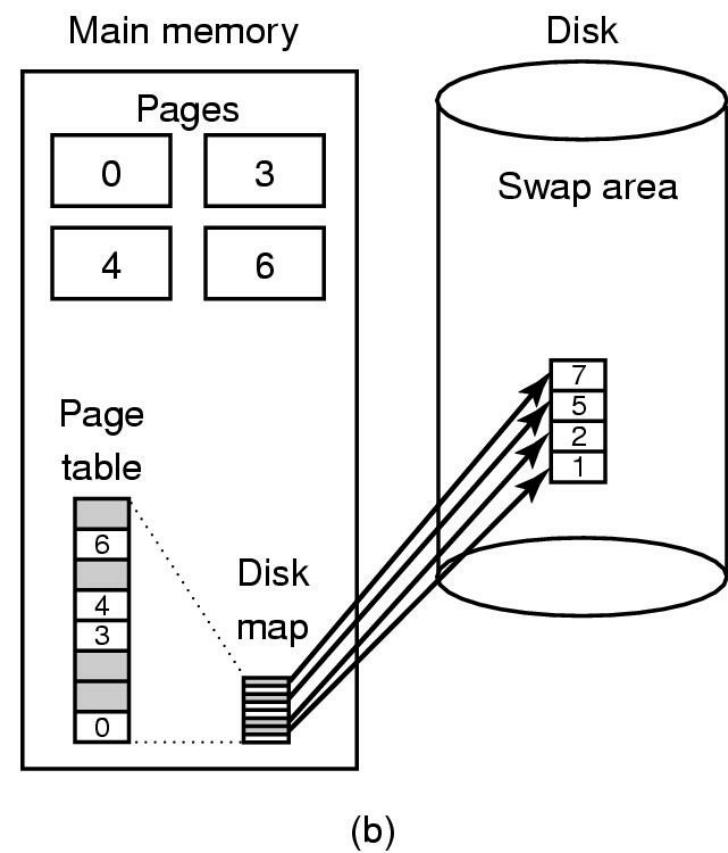
- Cada vez que se crea un proceso **se reserva una zona del área de intercambio** igual al tamaño de imagen del proceso. A cada proceso se le asigna la dirección en disco de su área de intercambio. La lectura se realiza sumando el número de página virtual a la dirección de comienzo del área asignada al proceso.
- No se asigna nada inicialmente.** A cada página se le asigna su espacio en disco cuando se va a intercambiar, y el espacio se libera cuando la página vuelve a memoria. Problema: se debe llevar contabilidad en memoria (página a página) de la localización de las páginas en disco.



Área de Intercambio (cont.)



(a)



(b)



Área de Intercambio (cont.)

- ✓ Cuando una página no esta en memoria, sino en swap, como podríamos saber en que parte del área de intercambio está?
 - ✓ Rta: El PTE de dicha pagina tiene el bit V=0 y todos los demás bits sin usar!



Área de Intercambio - Linux

- Permite definir un número predefinido de áreas de Swap
- swap_info es un arreglo que contiene estas estructuras

<linux/swap.h>

```
64 struct swap_info_struct {  
65     unsigned int flags;  
66     kdev_t swap_device;  
67     spinlock_t sdev_lock;  
68     struct dentry * swap_file;  
69     struct vfsmount *swap_vfsmnt;  
70     unsigned short * swap_map;  
71     unsigned int lowest_bit;  
72     unsigned int highest_bit;  
73     unsigned int cluster_next;  
74     unsigned int cluster_nr;  
75     int prio;  
76     int pages;  
77     unsigned long max;  
78     int next;  
79 };
```



Área de Intercambio - Linux (cont.)

- Cada área es dividida en un número fijo de slots según el tamaño de la página

- Cuando una página es llevada a disco, Linux utiliza el PTE para almacenar 2 valores:
 - ✓ En número de área
 - ✓ El desplazamiento en el área (24 bits, lo que limita el tamaño máximo del área a 64 Gb)



Área de Intercambio - Linux (cont.)

Figure 17-6. Swap area data structures

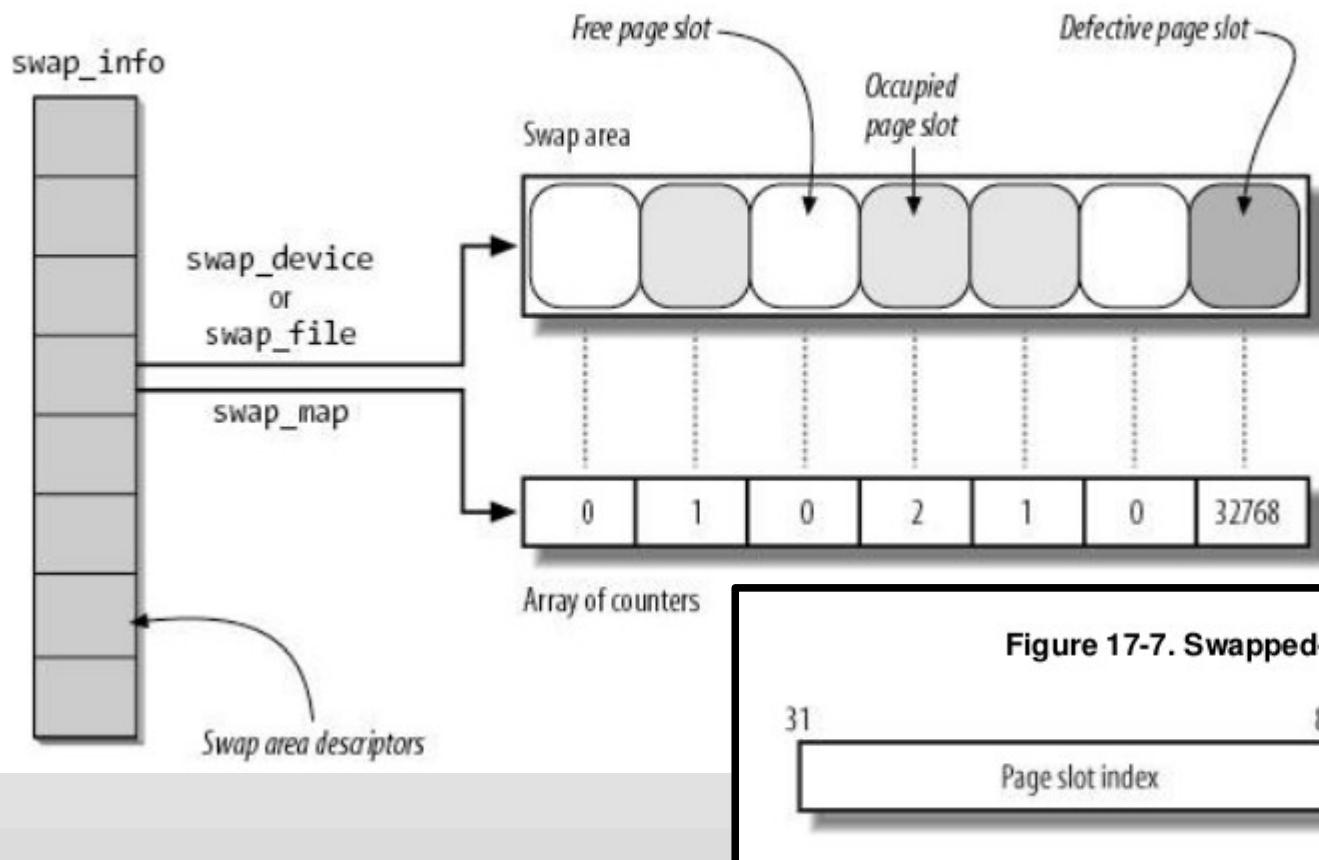
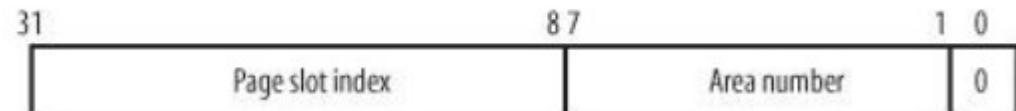


Figure 17-7. Swapped-out page identifier



Ref: Understanding the Linux Kernel <http://ceata.org/~tct/resource/utlk.pdf>



Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de
Memoria -Ejemplos



Versión: Septiembre 2023

Palabras Claves: Procesos, Espacio de Direcciones, Memoria Mapa de Memoria, Página, Memoria Virtual, Tablas de Páginas

Referencia: <https://juncotic.com/mapa-de-memoria-de-un-proceso-en-linux/>



Mapa de Memoria

- ✓ Un mapa de memoria, (memory map o memory layout) es una estructura de datos que indica al sistema operativo cómo está distribuida la memoria de un proceso, los segmentos que la componen, y los datos almacenados en cada uno de ellos.
- ✓ Cuando se ejecuta un programa en GNU/Linux, se crea un mapa de memoria en la que se carga variables inicializadas, variables no inicializadas, segmentos de memoria dinámica, la pila, y el código binario de la aplicación (o una parte de él).



Mapa de Memoria

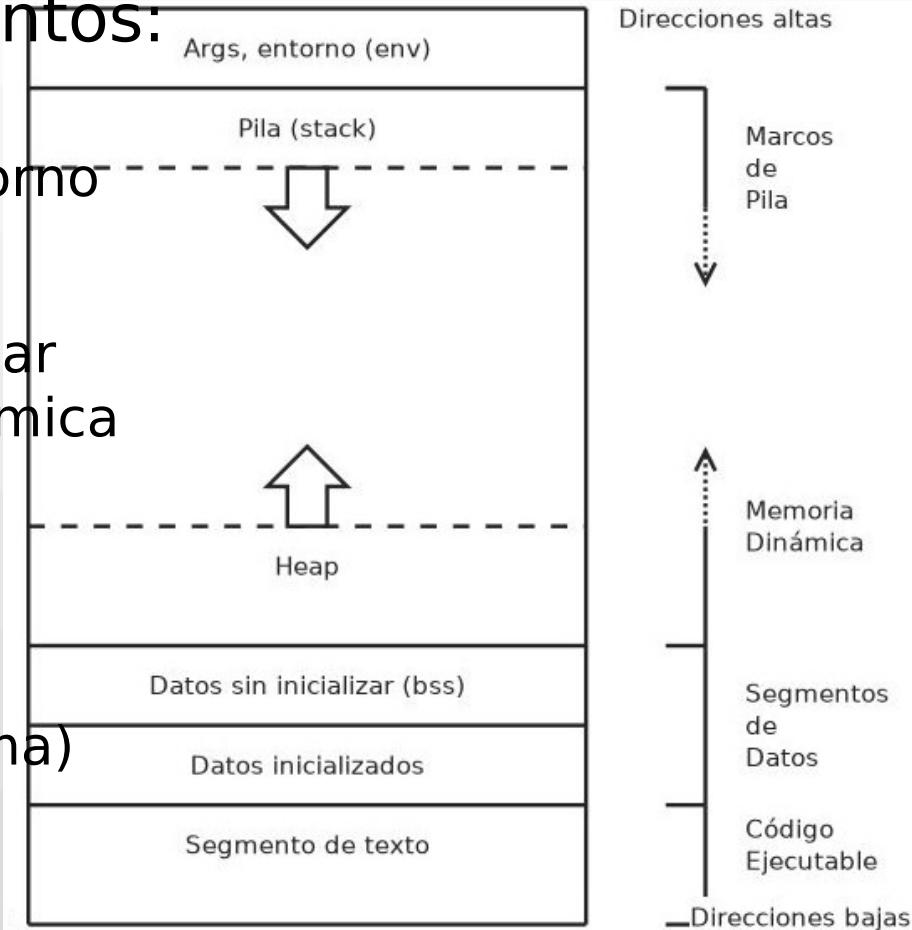
- ✓ GNU/Linux utiliza gestión de memoria basada en segmentación y paginación
- ✓ Las páginas de memoria son de tamaño fijo, por ejemplo, 4 KiB (podemos leer el tamaño de la página con el comando `getconf PAGESIZE`), mientras que la segmentación implica segmentos de tamaño variable.
- ✓ GNU/Linux divide el mapa de memoria de un proceso en segmentos de diferentes tamaños, cada uno dividido, a su vez, en páginas de tamaño fijo.



Mapa de Memoria

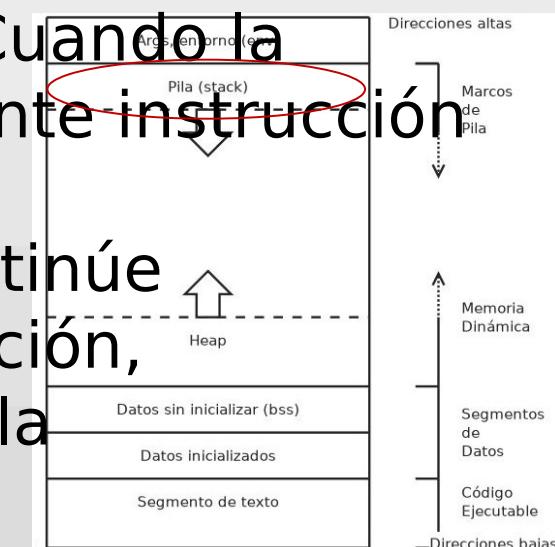
El mapa de memoria de un proceso se divide generalmente en 6 segmentos:

- Argumentos de la línea de comandos y variables de entorno
- Stack o pila del proceso.
- Heap o espacio para almacenar segmentos de memoria dinámica
- Datos no inicializados (BSS)
- Datos inicializados
- Segmento de texto (código binario de un programa)



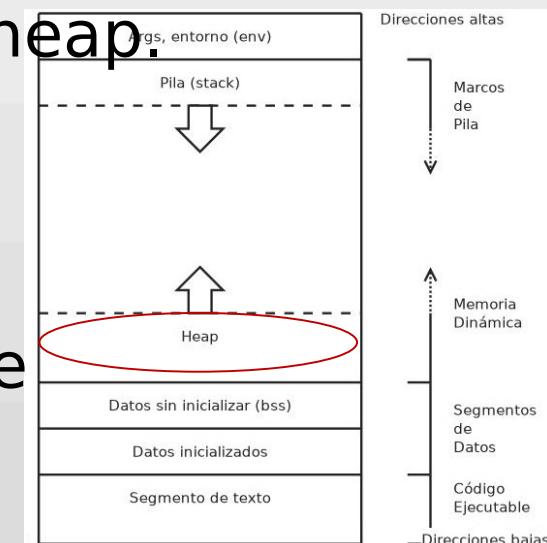
Mapa de Memoria - Stack

- ✓ En este segmento se almacenan las variables locales de las funciones, los argumentos pasados a cada función llamada, y los punteros de retorno de las mismas.
- ✓ Cuando se llama a una función se crea un marco de pila, se almacenan variables locales, argumentos y la dirección de retorno de la función. Cuando la función termina, se carga como siguiente instrucción la dirección de retorno, de modo que la ejecución continúe por donde iba antes de llamar a la función, y se procede a eliminar el marco de pila



Mapa de Memoria - Heap

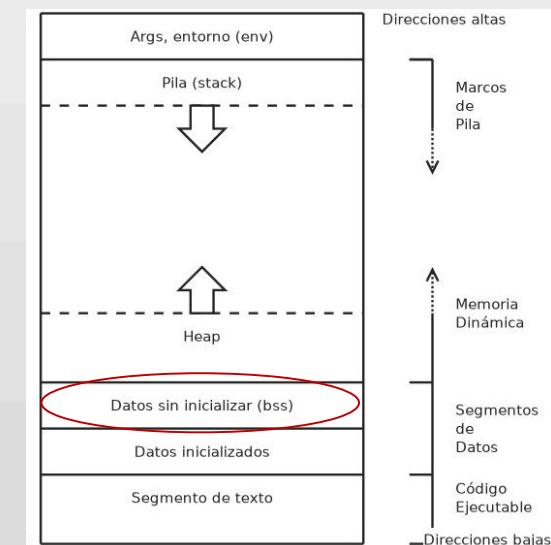
- En este segmento se almacena la memoria dinámica creada por el proceso (malloc(), calloc(), realloc() por ejemplo).
- Cuando se reserva una posición para memoria dinámica el programa adquiere dicho espacio desde el heap. Si se libera dicha memoria (free()), se reduce el espacio ocupado dentro del heap.
- Cuando se libera espacio de memoria dinámica se devuelve al heap, pero no a la memoria del sistema operativo, por lo que el heap puede que comience a fragmentarse.



Mapa de Memoria – Datos sin inicializar

Este segmento, también conocido como BSS (heredado de lenguaje ensamblador) almacena todas las variables globales y estáticas que no están inicializadas a cero o no tienen un valor de inicialización en el código fuente

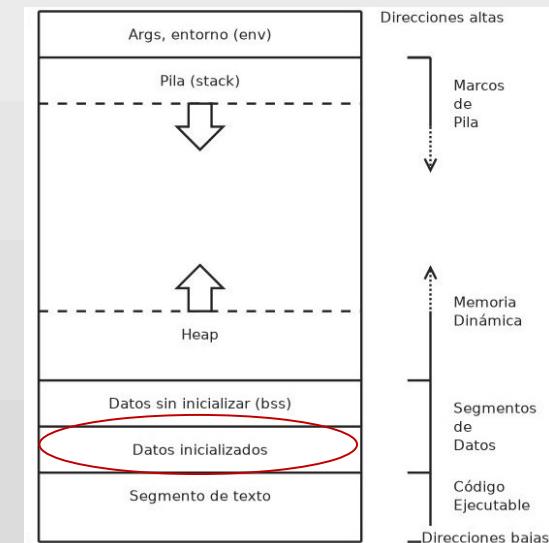
```
1 | int num; //global  
2 | static int x=0;
```



Mapa de Memoria – Datos inicializados

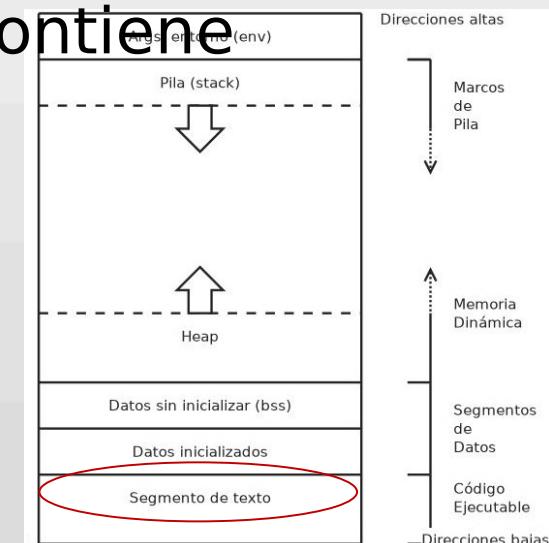
Contiene las variables globales y estáticas del programa, que a su vez fueron inicializadas con valores distintos de cero. Este segmento puede clasificarse como de sólo lectura y de lectura-escritura.

```
1 | char cadena[] = "Hola Mundo"; //global
2 | int contador = 1; //global
3 | const int num = 5; //global
```



Mapa de Memoria - Texto

- Este segmento almacena las instrucciones ejecutables del programa, por lo que también se lo denomina segmento de código.
- Almacena la representación en código de máquina de las instrucciones del programa. Este segmento en general puede ser compartido entre diferentes procesos, ya que no se modifica, y si contiene código de librerías compartidas no es necesario duplicarlo en memoria
- Es un segmento de sólo lectura



Mapa de Memoria - Ejemplos

- Para los siguientes ejemplos usaremos el comando size, que en su salida muestra el tamaño del segmento de texto, de datos inicializados, y el bss.
- Tomando como punto de partida el siguiente código fuente de C:

```
1 | #include<stdio.h>
2 |
3 | int main(){
4 |     return 0;
5 | }
```

- Compilando el código anterior con gcc y ejecutando el comando size:

```
diego@cryptos:~/tmp/a
$ gcc ejemplo.c -o ejecutable && size ejecutable
  text    data     bss   dec   hex filename
 1136      512       8   1656    678 ejecutable
diego@cryptos:~/tmp/a
$
```

- Se ve que el segmento de datos contiene 512 bytes, mientras que el bss solamente 8 bytes.



Mapa de Memoria - Ejemplos

- Para los siguientes ejemplos usaremos el comando size, que en su salida muestra el tamaño del segmento de texto, de datos inicializados, y el bss.

- Creemos ahora, por ejemplo, una variable global, de tipo double:

```
1 #include<stdio.h>
2 double numero;
3
4 int main(){
5     return 0;
6 }
```

- Comprobemos el resultado con el comando size:

```
diego@cryptos in /tmp/a
$ gcc ejemplo.c -o ejecutable && size ejecutable
text    data    bss    dec   hex filename
1136      512      16    1664    680 ejecutable
diego@cryptos in /tmp/a
$
```

- Se ve que bss aumentó 8 bytes. Esto es correcto puesto que la variable global no está inicializada, y una variable de tipo double en C ocupa 8 bytes en memoria.



Mapa de Memoria - Ejemplos

- Para los siguientes ejemplos usaremos el comando size, que en su salida muestra el tamaño del segmento de texto, de datos inicializados, y el bss.
- Veamos ahora qué pasa si la inicializamos con un valor distinto de cero.

```
1 #include<stdio.h>
2 double numero = 123;
3
4 int main(){
5     return 0;
6 }
```

- Comprobemos el resultado con el comando size:

```
diego@cryptos in /tmp/a
$ gcc ejemplo.c -o ejecutable && size ejecutable
text    data    bss    dec   hex filename
1136      520       8    1664    680 ejecutable
diego@cryptos in /tmp/a
$
```

- Aquí se ve que se redujo en 8 bytes el bss, ya no tenemos esa variable no inicializada. Sin embargo, se incrementó en 8 bytes el segmento de datos inicializados.

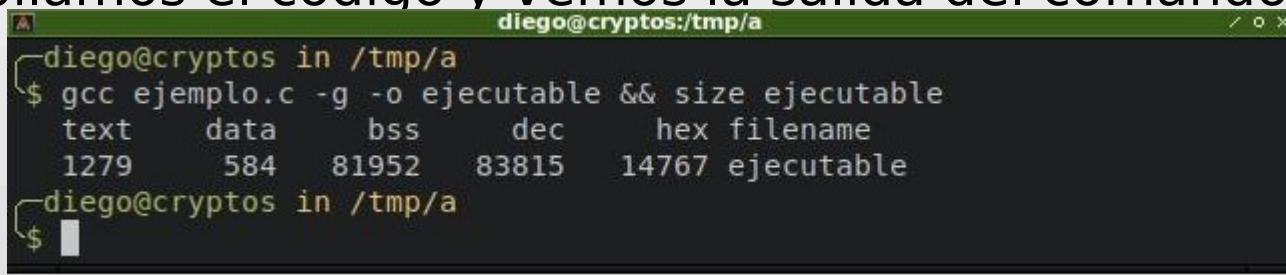


Mapa de Memoria - pmap

- ✓ Un interesante comando para leer el mapa de memoria de un proceso en ejecución es pmap

```
1 #include<stdio.h>
2
3 char cadena[8192*10];
4
5 int main(){
6     getchar();
7     return 0;
8 }
```

- ✓ Compilamos el código y vemos la salida del comando size:



A terminal window titled "diego@cryptos:~/tmp/a" showing the compilation of a C program and its memory usage analysis.

```
diego@cryptos in /tmp/a
$ gcc ejemplo.c -g -o ejecutable && size ejecutable
text      data      bss      dec      hex filename
1279      584      81952    83815    14767 ejecutable
diego@cryptos in /tmp/a
$
```

- ✓ Como puede observarse, el segmento de datos no inicializados (bss) contiene 80 KiB de datos, el arreglo del 80 KiB definido de manera global



Mapa de Memoria - pmap

- ✓ Si ejecutamos el código éste se detendrá en la penúltima línea, getchar(), esperando que presionemos una tecla. Si lo dejamos corriendo y en otra terminal ejecutamos el comando pmap pasando por argumento el PID de nuestro ejecutable, veremos

```
[diego@cryptos in /tmp/a
$ pmap -x $(pidof ejecutable )
48216: ./ejecutable
Address          Kbytes   RSS   Dirty Mode  Mapping
00005616523d9000    4      4       4 r---- executable
00005616523da000    4      4       4 r-x-- executable
00005616523db000    4      0       0 r---- executable
00005616523dc000    4      4       4 r---- executable
00005616523dd000    4      4       4 rw--- executable
00005616523de000   80      0       0 rw--- [ anon ]
00005616532a5000  132      4       4 rw--- [ anon ]
00007f1cfb6c4000     8      4       4 rw--- [ anon ]
00007f1cfb6c6000  136    136      0 r---- libc.so.6
00007f1cfb6e8000  1384    672      0 r-x-- libc.so.6
00007f1cfb842000   352      64      0 r---- libc.so.6
00007f1cfb89a000    16      16      16 r---- libc.so.6
00007f1cfb89e000     8      8       8 rw--- libc.so.6
00007f1cfb8a0000    60      28      28 rw--- [ anon ]
00007f1cfb8f3000     4      4       0 r---- ld-linux-x86-64.so.2
00007f1cfb8f4000  152    152      0 r-x-- ld-linux-x86-64.so.2
00007f1cfb91a000    40      40      0 r---- ld-linux-x86-64.so.2
00007f1cfb924000     8      8       8 r---- ld-linux-x86-64.so.2
00007f1cfb926000     8      8       8 rw--- ld-linux-x86-64.so.2
00007ffffe684b000  136     16      16 rw--- [ stack ]
00007ffffe69c4000   16      0       0 r---- [ anon ]
00007ffffe69c8000     8      4       0 r-x-- [ anon ]
ffffffffff600000     4      0       0 --x-- [ anon ]
-----
total kB        2572    1180    108
```

- ✓ Se llama anónima a la memoria mapeada no asociada a archivos en el disco
- ✓ La línea marcada indica que tenemos un mapeo anónimo de 80 KiB.
- ✓ Columnas:
 - ✓ Address: la dirección de inicio de la posición de memoria en cuestión.
 - ✓ Kbytes: tamaño de esa región en KiB.
 - ✓ RSS: Resident set size, parte de la memoria realmente en RAM.
 - ✓ Dirty: estado de las páginas de memoria.
 - ✓ Mode: permisos de acceso a esa región de memoria
 - ✓ Mapping: el nombre de la app o librería asociada a esa región de memoria.



Mapa de Memoria - `pmap`

- ✓ Inicialicemos la memoria con un valor, y analicemos la salida de `size` y de `pmap` nuevamente.

```
1 #include<stdio.h>
2
3 char cadena[8192*10] = "hola";
4
5 int main(){
6     getchar();
7     return 0;
8 }
```

- ✓ Compilamos el código y vemos la salida del comando `size`:



```
diego@cryptos in /tmp/a
$ gcc ejemplo.c -g -o ejecutable && size ejecutable
  text    data     bss      dec      hex filename
 1279    82520      8    83807    1475f ejecutable
diego@cryptos in /tmp/a
$
```

- ✓ El comando `size` nos muestra que los 80 KiB que antes estaban en el segmento `bss` ahora han pasado al segmento de datos inicializados.



Mapa de Memoria - pmap

- ✓ La salida de pmap muestra que ahora el segmento ya no es anónimo, ha pasado a datos inicializados, y se sumó a los 4 KiB

Address	Kbytes	RSS	Dirty	Mode	Mapping
000055688033b000	4	4	4	r----	ejecutable
000055688033c000	4	4	4	r-x--	ejecutable
000055688033d000	4	0	0	r----	ejecutable
000055688033e000	4	4	4	r----	ejecutable
000055688033f000	84	64	64	rw---	ejecutable
00005568818b6000	132	4	4	rw---	[anon]
00007fa150bad000	8	4	4	rw---	[anon]
00007fa150baf000	136	132	0	r----	libc.so.6
00007fa150bd1000	1384	700	0	r-x--	libc.so.6
00007fa150d2b000	352	64	0	r----	libc.so.6
00007fa150d83000	16	16	16	r----	libc.so.6
00007fa150d87000	8	8	8	rw---	libc.so.6
00007fa150d89000	60	28	28	rw---	[anon]
00007fa150ddc000	4	4	0	r----	ld-linux-x86-64.so.2
00007fa150ddd000	152	152	0	r-x--	ld-linux-x86-64.so.2
00007fa150e03000	40	40	0	r----	ld-linux-x86-64.so.2
00007fa150e0d000	8	8	8	r----	ld-linux-x86-64.so.2
00007fa150e0f000	8	8	8	rw---	ld-linux-x86-64.so.2
00007ffc9e5ff000	136	20	20	rw---	[stack]
00007ffc9e658000	16	0	0	r----	[anon]
00007ffc9e65c000	8	4	0	r-x--	[anon]
ffffffffffff600000	4	0	0	--x--	[anon]
total kB	2572	1268	172		

✓ Columnas:

- ✓ Address: la dirección de inicio de la posición de memoria en cuestión.
- ✓ Kbytes: tamaño de esa región en KiB.
- ✓ RSS: Resident set size, parte de la memoria realmente en RAM.
- ✓ Dirty: estado de las páginas de memoria.
- ✓ Mode: permisos de acceso a esa región de memoria por parte del proceso.
- ✓ Mapping: el nombre de la app o librería asociada a esa región de memoria.



Introducción a los Sistemas Operativos

Subsistema de
Entrada / Salida



Versión: Mayo 2020

Palabras Claves: Metas, Aspectos de dispositivos, Subsistema de IO

Algunas diapositivas/ímágenes han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos), el de Silberschatz (Operating Systems Concepts) y Tenembaum (Sistemas Operativos Modernos 3er Edición). También se incluyen diapositivas cedidas por Microsoft S.A.



Responsabilidades del SO

Controlar dispositivos de E/S

- Generar comandos
- Manejar interrupciones
- Manejar errores

Proporcionar una interfaz de utilización



Problemas

- Heterogeneidad de dispositivos
- Características de los dispositivos
- Velocidad
- Nuevos tipos de dispositivos
- Diferentes formas de realizar E/S (ver anexo)



Aspectos de los dispositivos de I/O

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read/write	CD-ROM graphics controller disk



Aspectos de los dispositivos de I/O (cont)

Unidad de Transferencia

✓ Dispositivos por bloques (discos):

- ◆ Operaciones: Read, Write, Seek

✓ Dispositivos por Caracter (keyboards, mouse, serial ports)

- ◆ Operaciones: get, put

Formas de Acceso

✓ Secuencial o Aleatorio



Aspectos de los dispositivos de I/O (cont)

✓ Tipo de acceso

- Acceso Compartido: Disco Rígido
- Acceso Exclusivo: Impresora

✓ Tipo de acceso:

- Read only: CDROM
- Write only: Pantalla
- Read/Write: Disco



Aspectos de los dispositivos de I/O (cont)

Velocidad

Dispositivo	Velocidad de transferencia de datos
Teclado	10 bytes/seg
Ratón	100 bytes/seg
Módem de 56K	7 KB/seg
Escáner	400 KB/seg
Cámara de video digital	3.5 MB/seg
802.11g inalámbrico	6.75 MB/seg
CD-ROM de 52X	7.8 MB/seg
Fast Ethernet	12.5 MB/seg
Tarjeta Compact Flash	40 MB/seg
FireWire (IEEE 1394)	50 MB/seg
USB 2.0	60 MB/seg
Red SONET OC-12	78 MB/seg
Disco SCSI Ultra 2	80 MB/seg
Gigabit Ethernet	125 MB/seg
Unidad de disco SATA	300 MB/seg
Cinta de Ultrium	320 MB/seg
Bus PCI	528 MB/seg



Metas, Objetivos y Servicios

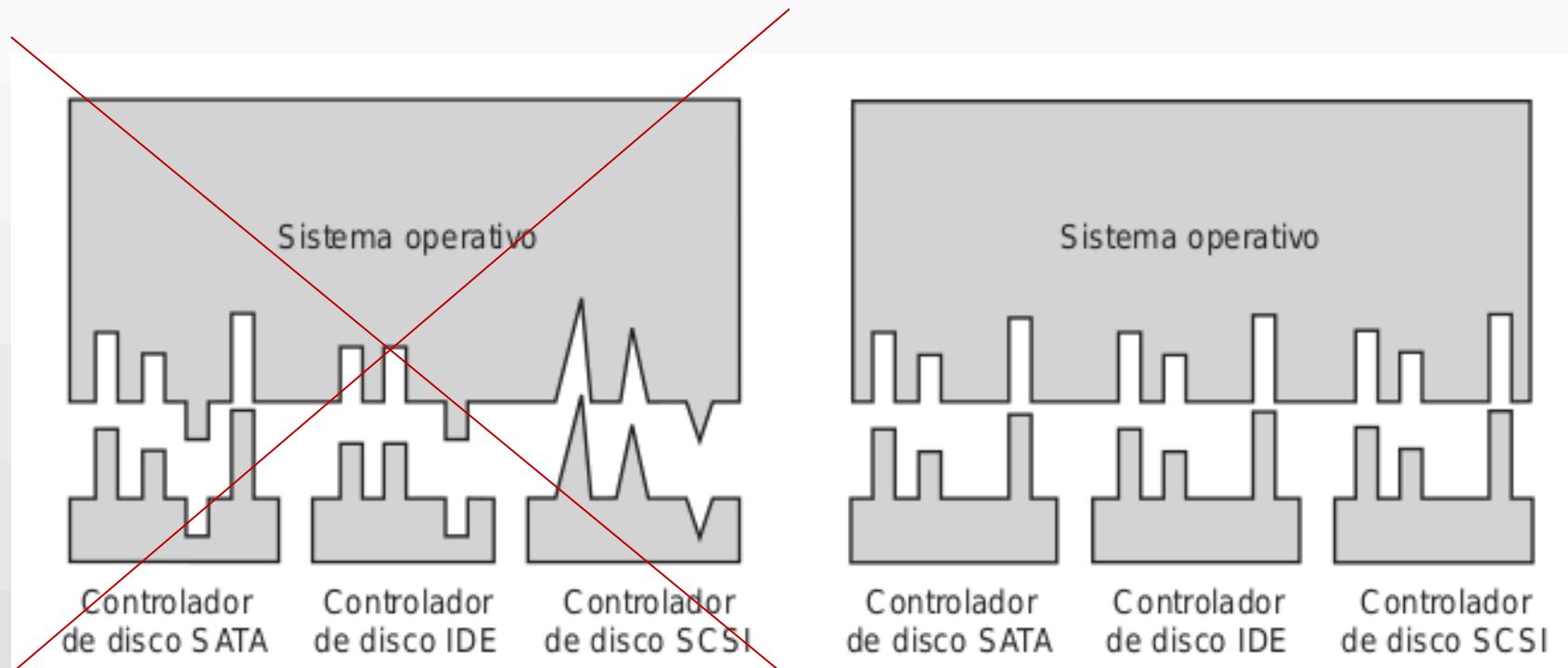
Generalidad:

- ✓ Es deseable manejar todos los dispositivos de I/O de una manera uniforme, estandarizada
- ✓ Ocultar la mayoría de los detalles del dispositivo en las rutinas de niveles más “bajos” para que los procesos vean a los dispositivos, en términos de operaciones comunes como: read, write, open, close, lock, unlock



Metas, Objetivos y Servicios

Interfaz Uniforme



Metas, Objetivos y Servicios

Eficiencia

- ✓ Los dispositivos de I/O pueden resultar extremadamente lentos respecto a la memoria y la CPU
- ✓ El uso de la multi-programación permite que un procesos espere por la finalización de su I/O mientras que otro proceso se ejecuta



Metas, Objetivos y Servicios

Planificación

- ✓ Organización de los requerimientos a los dispositivos
- ✓ Ej: Planificación de requerimientos a disco para minimizar tiempos



Metas, Objetivos y Servicios

- Buffering – Almacenamiento de los datos en memoria mientras se transfieren
 - ✓ Solucionar problemas de velocidad entre los dispositivos
 - ✓ Solucionar problemas de tamaño y/o forma de los datos entre los dispositivos



Metas, Objetivos y Servicios

- ✓ Caching - Mantener en memoria copia de los datos de reciente acceso para mejorar performance

- ✓ Spooling - Administrar la cola de requerimientos de un dispositivo
 - ✓ Algunos dispositivos de acceso exclusivo, no pueden atender distintos requerimientos al mismo tiempo: Por ej. Impresora
 - ✓ Spooling es un mecanismo para coordinar el acceso concurrente al dispositivo



Metas, Objetivos y Servicios

- ✓ Reserva de Dispositivos: Acceso exclusivo
- ✓ Manejo de Errores:
 - ✓ El S.O. debe administrar errores ocurridos (lectura de un disco, dispositivo no disponible, errores de escritura)
 - ✓ La mayoría retorna un número de error o código cuando la I/O falla.
 - ✓ Logs de errores



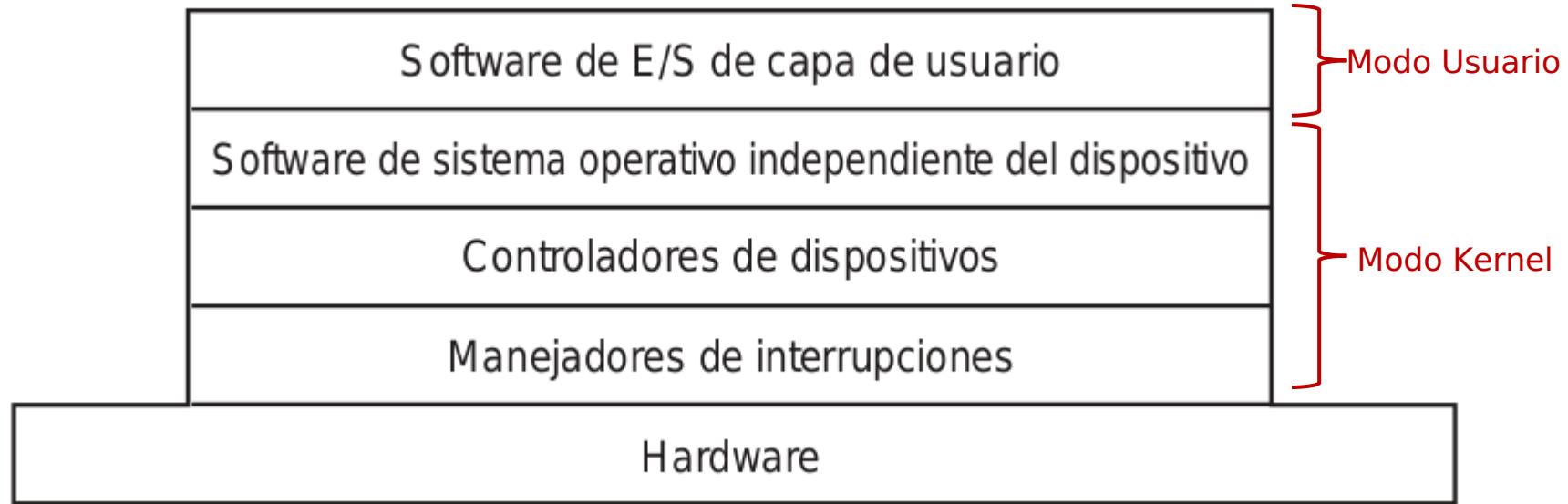
Metas, Objetivos y Servicios

✓ Formas de realizar I/O

- ✓ Bloqueante: El proceso se suspende hasta que el requerimiento de I/O se completa
 - ◆ Fácil de usar y entender
 - ◆ No es suficiente bajo algunas necesidades
- ✓ No Bloqueante: El requerimiento de I/O retorna en cuanto es posible
 - ◆ Ejemplo: Interfaz de usuario que recibe input desde el teclado/mouse y se muestra screen.
 - ◆ Ejemplo: Aplicación de video que lee frames desde un archivo mientras va mostrandolo en pantalla.



Diseño



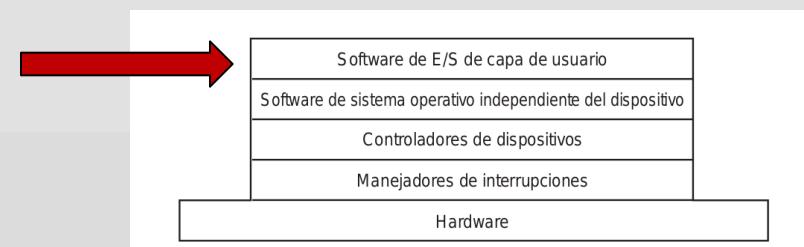
Diseño - Software capa de usuario

Librerías de funciones

- Permiten acceso a SysCalls
- Implementan servicios que no dependen del Kernel

Procesos de apoyo

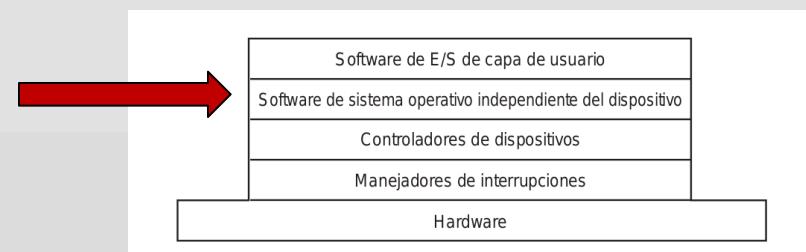
- Demonio de Impresión (spooling)



Diseño - Software independiente SO

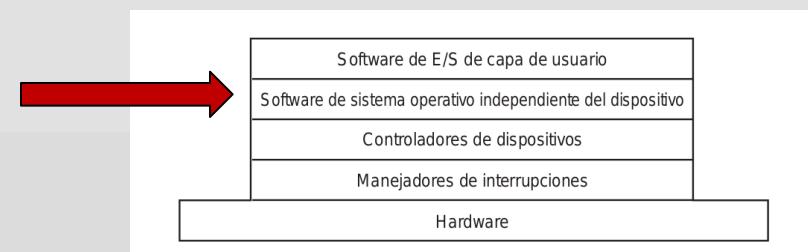
Brinda los principales servicios de E/S antes vistos

- Interfaz uniforme
- Manejo de errores
- Buffer
- Asignación de Recursos
- Planificación



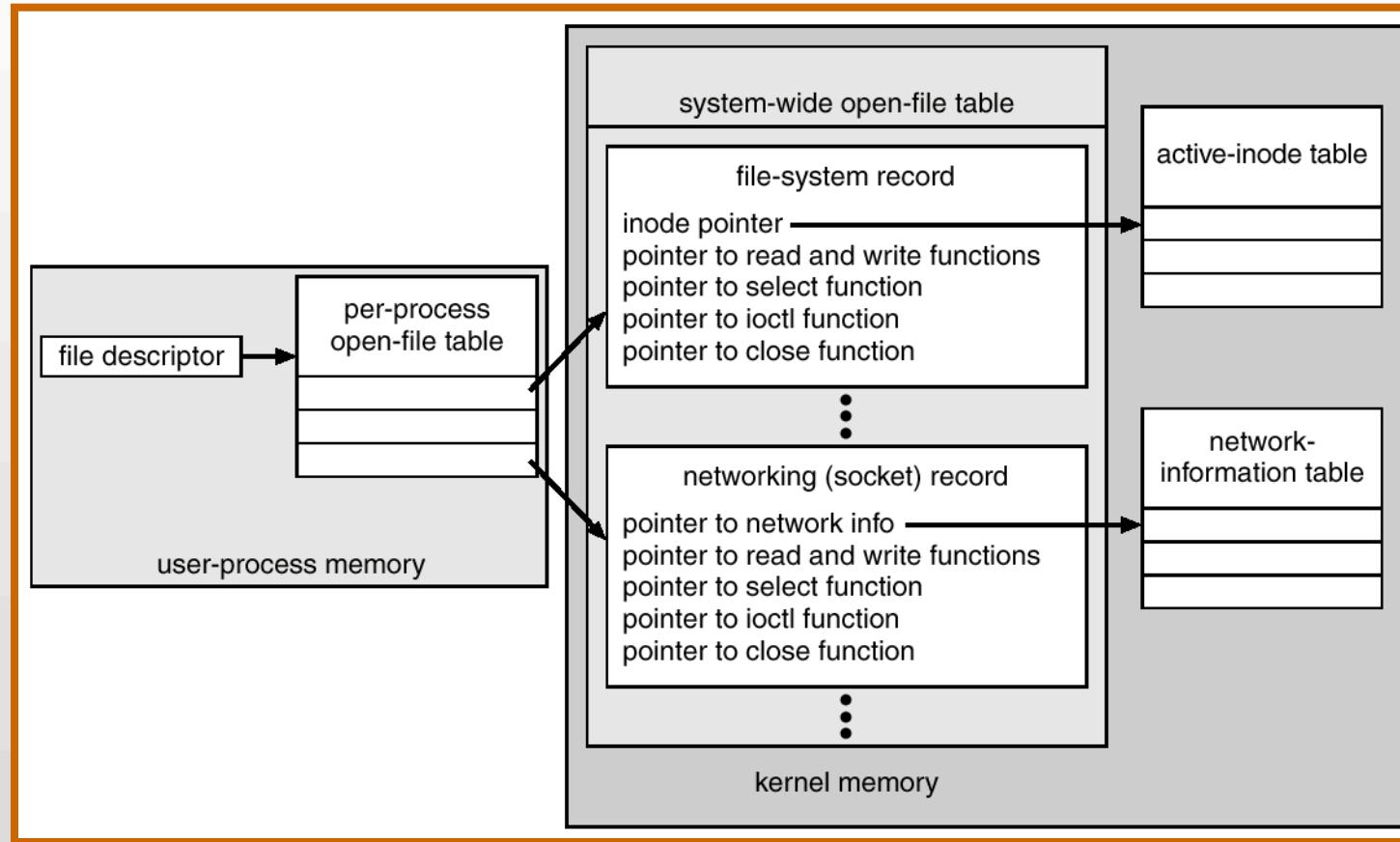
Diseño - Software independiente SO

- El Kernel mantiene la información de estado de cada dispositivo o componente
 - ✓ Archivos abiertos
 - ✓ Conexiones de red
 - ✓ Etc.
- Hay varias estructuras complejas que representan buffers, utilización de la memoria, disco, etc.



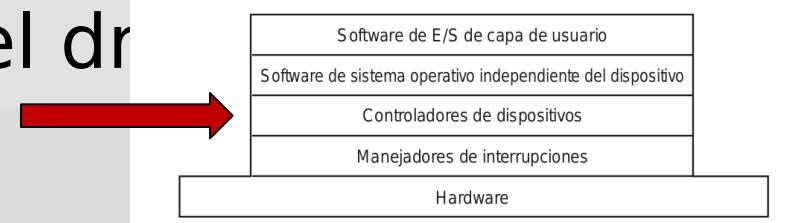
Diseño - Software independiente SO

UNIX I/O Kernel Structure



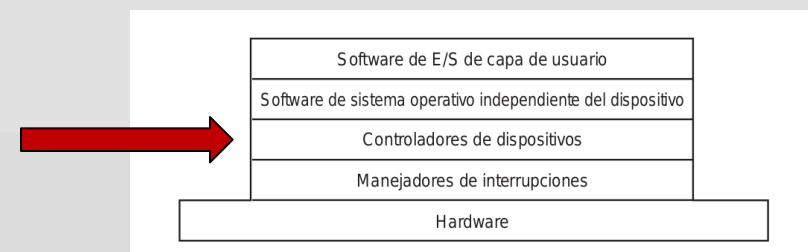
Diseño - Controladores (Drivers)

- Contienen el código dependiente del dispositivo
- Manejan un tipo dispositivo
- Traducen los requerimientos abstractos en los comandos para el dispositivo
 - ✓ Escribe sobre los registros del controlador
 - ✓ Acceso a la memoria mapeada
 - ✓ Encola requerimientos
- Comúnmente las interrupciones generadas por los dispositivos son atendidas por funciones provistas por el driver



Diseño - Controladores (Drivers)

- Interfaz entre el SO y el HARD
- Forman parte del espacio de memoria del Kernel
 - ✓ En general se cargan como módulos
- Los fabricantes de HW implementan el driver en función de una API especificada por el SO
 - ✓ open(), close(), read(), write(), etc
- Para agregar nuevo HW sólo basta indicar el driver correspondiente sin necesidad de cambios en el Kernel



Driver - Ejemplo en Linux

Linux distingue 3 tipos de dispositivos

- ✓ Carácter: I/O programada o por interrupciones
- ✓ Bloque: DMA
- ✓ Red: Ports de comunicaciones

Los Drivers se implementan como módulos

- ✓ Se cargan dinámicamente

Debe tener al menos estas operaciones:

- ✓ `init_module`: Para instalarlo
- ✓ `cleanup_module`: Para desinstalarlo.



Driver - Ejemplo en Linux (cont.)

Operaciones que debe contener para I/O

- ✓ **open**: abre el dispositivo
- ✓ **release**: cerrar el dispositivo
- ✓ **read**: leer bytes del dispositivo
- ✓ **write**: escribir bytes en el dispositivo
- ✓ **ioctl**: orden de control sobre el dispositivo



Driver - Ejemplo en Linux (cont.)

Otras operaciones menos comunes

- ✓ `lseek`: posicionar el puntero de lectura/escritura
- ✓ `flush`: volcar los búferes al dispositivo
- ✓ `poll`: preguntar si se puede leer o escribir
- ✓ `mmap`: mapear el dispositivo en memoria
- ✓ `fsync`: sincronizar el dispositivo
- ✓ `fasync`: notificación de operación asíncrona
- ✓ `lock`: reservar el dispositivo
- ✓



Driver - Ejemplo en Linux (cont.)

- ✓ Por convención, los nombres de las operaciones comienzan con el nombre del dispositivo
- ✓ Por ejemplo, para `/dev/ptr`

```
int ptr_open (struct inode *inode, struct file *filp)

void ptr_release (struct inode *inode, struct file *filp)

ssize_t ptr_read (struct file *flip, char *buff,
                  size_t count, loff_t *offp)

ssize_t ptr_write (struct file *filp, const char *buff,
                   size_t count, loff_t *offp)

int ptr_ioctl (struct inode *inode, struct file *filp,
               unsigned int cmd, unsigned long arg)
```



Driver - Ejemplo en Linux (cont.)

Acceso al hardware

- ✓ Funciones para acceso a los puertos de I/O `<asm/io.h>`

```
unsigned char inb (unsigned short int port)
void outb (unsigned char value, unsigned short int port)
```

Leen o Escriben un byte en el puerto de E/S indicado

En MS-DOS

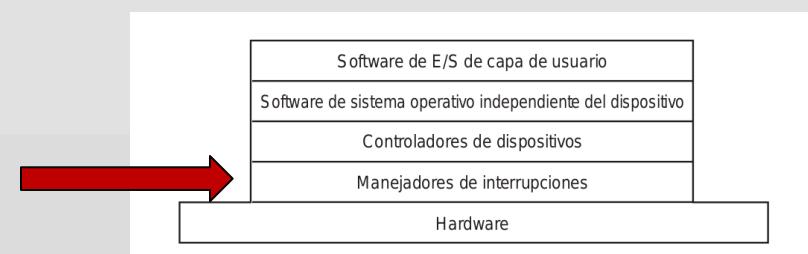
o 70 02
i 71
<retorna los minutos>

o 70 00
i 71
<retorna los segundos>

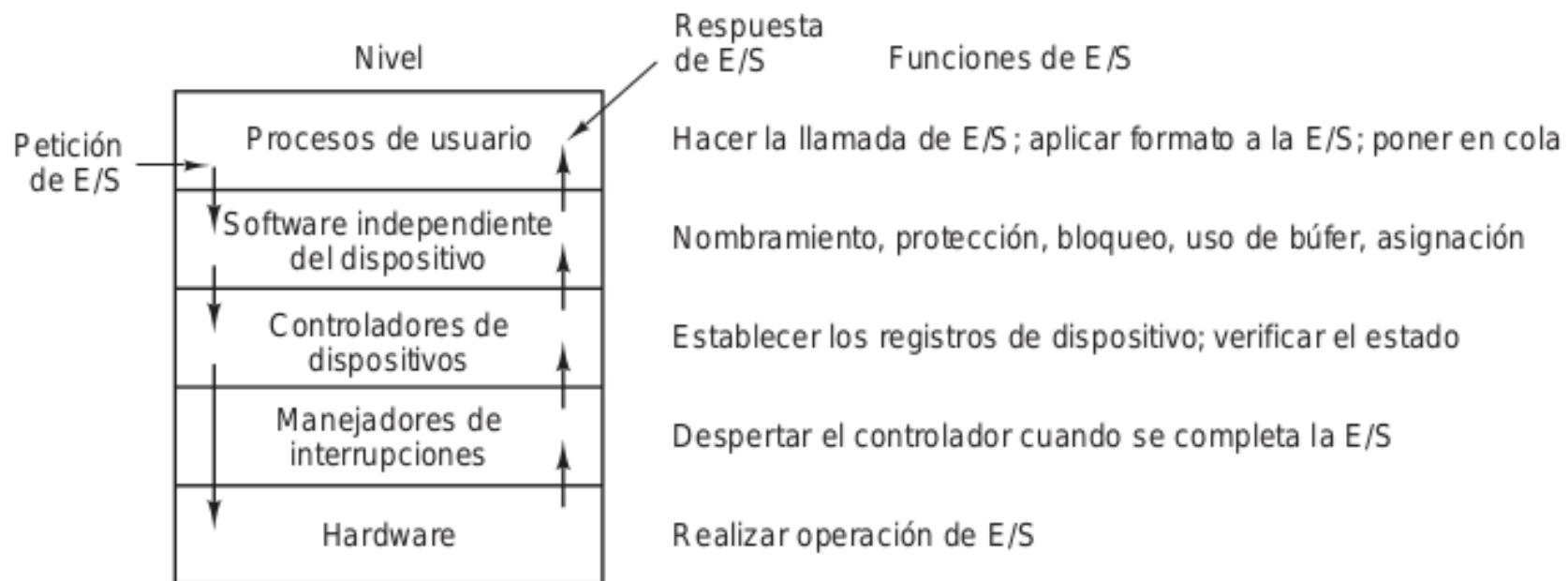


Diseño - Gestor de interrupciones

- Atiende todas las interrupciones del HW
- Deriva al driver correspondiente según interrupción
- Resguarda información
- Independiente del Driver



Ciclo de atención de un Requerimiento



Desde el Requerimiento de I/O hasta el Hardware

Consideremos la lectura sobre un archivo en un disco:

- ✓ Determinar el dispositivo que almacena los datos
 - Traducir el nombre del archivo en la representación del dispositivo.
- ✓ Traducir requerimiento abstracto en bloques de disco (Filesystem)
- ✓ Realizar la lectura física de los datos (bloques) en la memoria
- ✓ Marcar los datos como disponibles al proceso que realizó el requerimiento
 - Desbloquearlo
- ✓ Retornar el control al proceso



Desde el Requerimiento de I/O hasta el Hardware

```
nico@yoko:~$ ls -l /dev/sd*
```

```
brw-rw---- 1 root disk 8,  0 oct 28 11:32 /dev/sda
brw-rw---- 1 root disk 8,  1 oct 28 11:32 /dev/sda1
brw-rw---- 1 root disk 8,  2 oct 28 11:32 /dev/sda2
brw-rw---- 1 root disk 8,  5 oct 28 11:32 /dev/sda5
brw-rw---- 1 root disk 8, 16 oct 28 15:49 /dev/sdb
brw-rw---- 1 root disk 8, 17 oct 28 15:49 /dev/sdb1
```

```
nico@yoko:~$ █
```

```
nico@yoko:~$ cat /proc/devices
```

Character devices:

```
1 mem
4 /dev/vc/0
4 tty
4 ttys
5 /dev/tty
5 /dev/console
5 /dev/ptmx
5 ttyprintk
```

Block devices:

```
1 ramdisk
259 blkext
7 loop
8 sd
9 md
11 sr
65 sd
66 sd
67 cd
```

THE I/O SUBSYSTEM

block device switch table			
entry	open	close	strategy
0	gdopen	gdclose	gdstrategy
1	gtopen	gtclose	gtstrategy

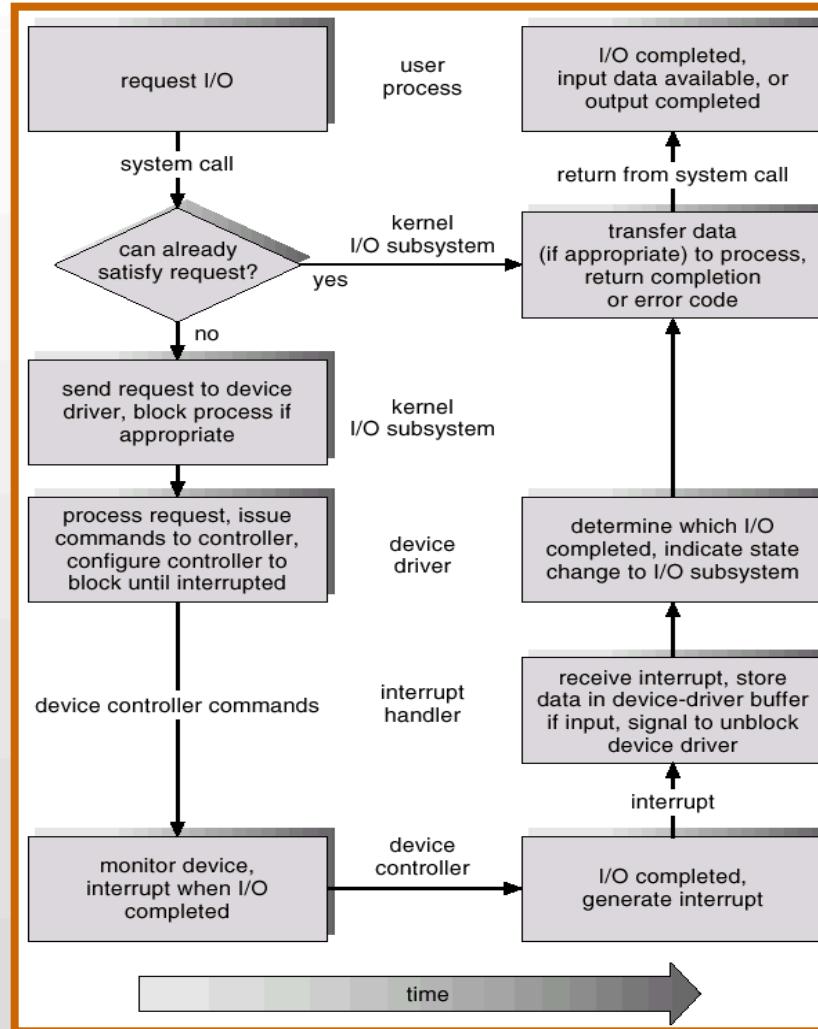
character device switch table

entry	open	close	read	write	ioctl
0	conopen	conclose	conread	conwrite	conioctl
1	dzbopen	dzbclose	dzbread	dzbwrite	dzbioctl
2	syopen	nulldev	syread	sywrite	syioctl
3	nulldev	nulldev	mmread	mmwrite	nodev
4	gdopen	gdclose	gdread	gdwrite	nodev
5	gtopen	gtclose	gtread	gtwrite	nodev

Figure 10.2. Sample Block and Character Device Switch Tables



Ciclo de vida de un requerimiento de I/O



Performance

- I/O es uno de los factores que mas afectan a la performance del sistema:
 - ✓ Utiliza mucho la CPU para ejecutar los drivers y el codigo del subsistema de I/O
 - ✓ Provoca Context switches ante las interrupciones y bloqueos de los procesos
 - ✓ Utiliza el bus de mem. en copia de datos:
 - Aplicaciones (espacio usuario) - Kernel
 - Kernel (memoria fisica) - Controladora



Mejorar la Performance

- Reducir el número de context switches
- Reducir la cantidad de copias de los datos mientras se pasan del dispositivo a la aplicación
- Reducir la frecuencia de las interrupciones, utilizando:
 - Transferencias de gran cantidad de datos
 - Controladoras mas inteligentes
 - Polling, si se minimiza la espera activa.
- Utilizar DMA



Introducción a los Sistemas Operativos

Anexo I Arquitectura de Entrada/Salida



- Versión: Octubre 2017
- Palabras Claves: Dispositivos de IO, Hardware de IO, IO programada, Polling, Interrupciones, DMA

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Variedad en los dispositivos de I/O

Legible para el usuario

- ✓ Usados para comunicarse con el usuario
 - ◆ Impresoras, Terminales: Pantalla, Teclado, Mouse

Legible para la máquina

- ✓ Utilizados para comunicarse con los componentes electrónicos
 - ◆ Discos, Cintas, Sensores, etc.

Comunicación

- ✓ Usados para comunicarse con dispositivos remotos
 - ◆ Líneas Digitales, Modems, Interfaces de red, etc.



Problemas que surgen

Amplia Variedad

- ✓ Manejan diferentes cantidad de datos
- ✓ En Velocidades Diferentes
- ✓ En Formatos Diferentes

La gran mayoría de los dispositivos de E/S son más lentos que la CPU y la RAM

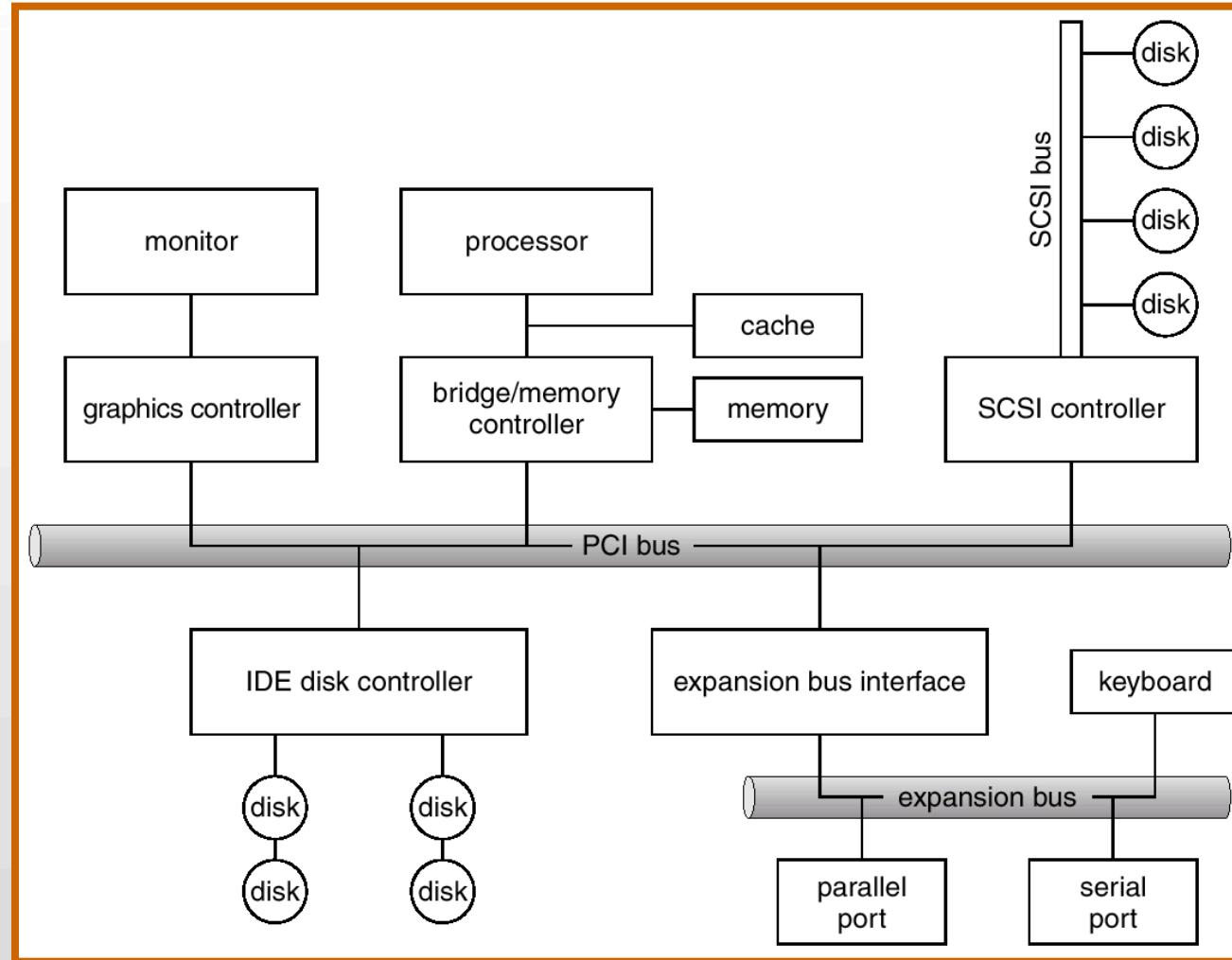


Hardware y software involucrado

- Buses
- Controladores
- Dispositivos
- Puertos de E/S - Registros
- Drivers
- Comunicación con controlador del dispositivo: I/O Programada, Interrupciones, DMA



Estructura de Bus de una PC



Comunicación: CPU - Controladora

- ¿Cómo puede la CPU ejecutar comandos o enviar/recibir datos de una controladora de un dispositivo?
 - ✓ La controladora tiene uno o mas registros:
 - Registros para señales de control
 - Registros para datos
- La CPU se comunica con la controladora escribiendo y leyendo en dichos registros



Comandos de I/O

CPU emite direcciones

- ✓ Para identificar el dispositivo

CPU emite comandos

- ✓ Control – Que hacer?

- ◆ Ej. Girar el disco

- ✓ Test – Controlar el estado

- ◆ Ej. power? Error?

- ✓ Read/Write

- ◆ Transferir información desde/hacia el dispositivo



Mapeo de E/S y E/S aislada

Correspondencia en memoria (Memory mapped I/O)

- ✓ Dispositivos y memoria comparten el espacio de direcciones.
- ✓ I/O es como escribir/leer en la memoria.
- ✓ No hay instrucciones especiales para I/O
 - ◆ Ya se dispone de muchas instrucciones para la memoria

Isolated I/O (Aislada, uso de Puertos de E/S)

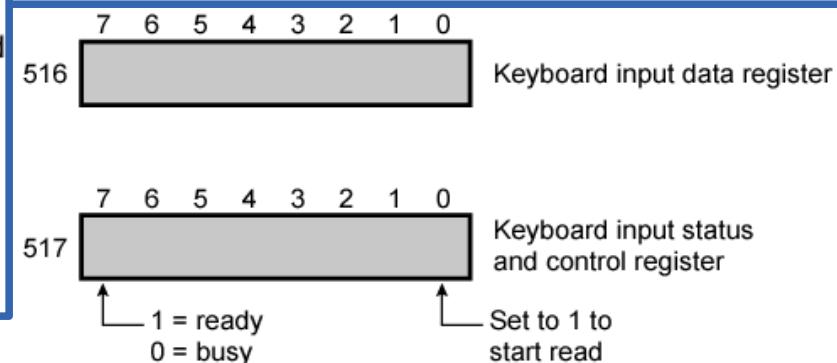
- ✓ Espacio separado de direcciones
- ✓ Se necesitan líneas de I/O. Puertos de E/S
- ✓ Instrucciones especiales
 - ◆ Conjunto Limitado



Memory Mapped and Isolated I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

En MS-DOS

```
o 70 02
i 71
<retorna los minutos>

o 70 00
i 71
<retorna los segundos>
```



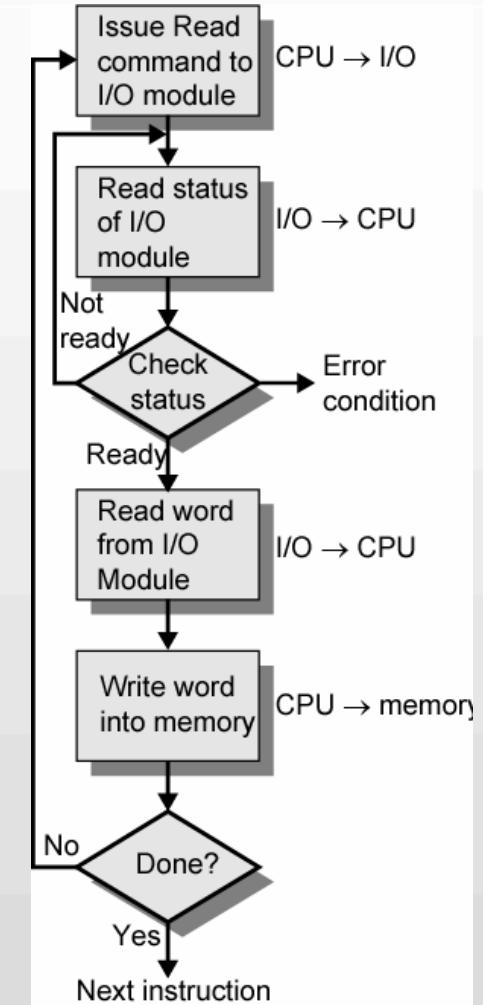
Técnicas de I/O - Programada

CPU tiene control directo sobre la I/O

- ✓ Controla el estado
- ✓ Comandos para leer y escribir
- ✓ Transfiere los datos

CPU espera que el componente de I/O complete la operación

Se desperdician ciclos de CPU



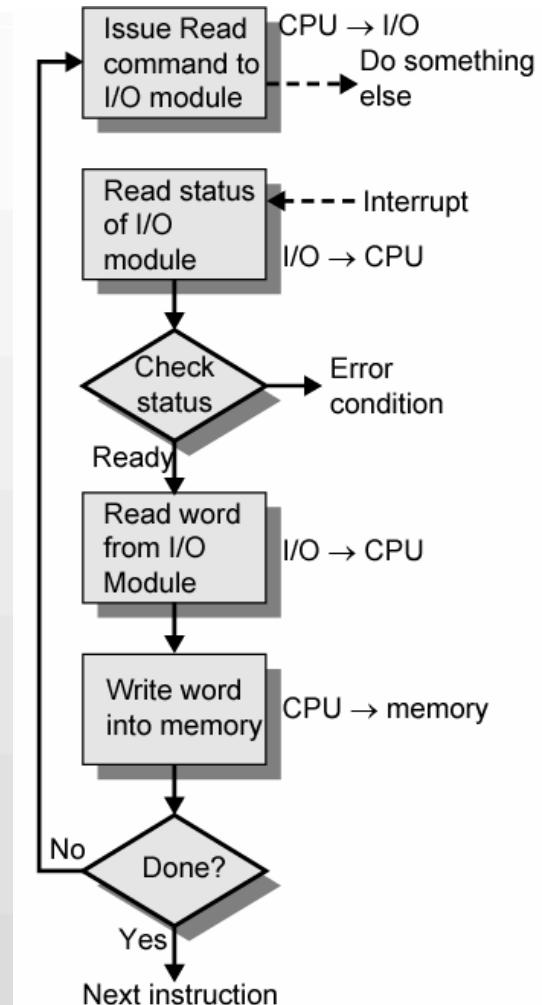
Polling

- ✓ En la I/O Programada, es necesario hacer polling del dispositivo para determinar el estado del mismo
 - ✓ Listo para recibir comandos
 - ✓ Ocupado
 - ✓ Error
- ✓ Ciclo de “Busy-wait” para realizar la I/O
- ✓ Puede ser muy costoso si la espera es muy larga



Técnicas de I/O - Manejada por Interrupciones

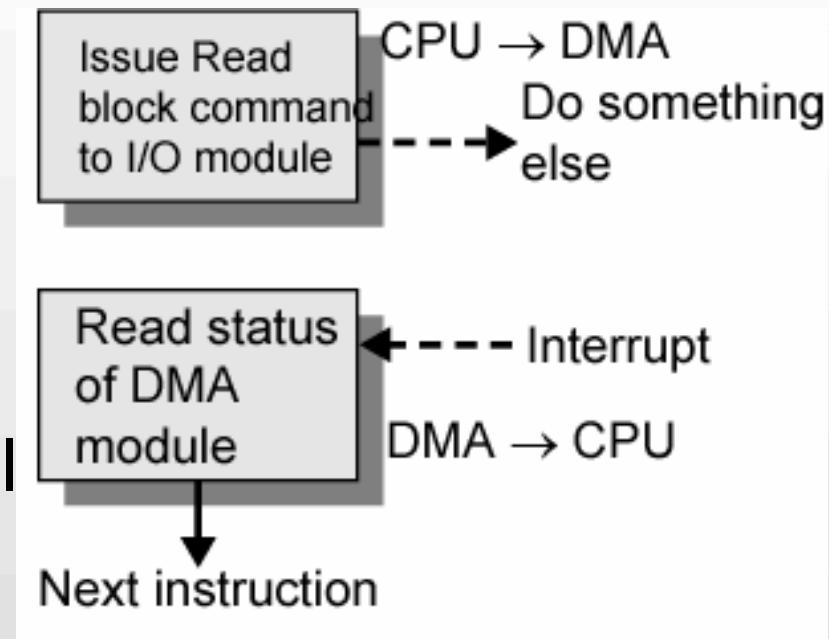
- Soluciona el problema de la espera de la CPU
- La CPU no repite el chequeo sobre el dispositivo
- El procesador continúa la ejecución de instrucciones
- El componente de I/O envía una interrupción cuando termina



Técnicas de I/O - DMA

DMA (Direct Memory Access)

- ✓ Un componente de DMA controla el intercambio de datos entre la memoria principal y el dispositivo
- ✓ El procesador es interrumpido luego de que el bloque entero fue transferido.



Pasos para una transferencia DMA

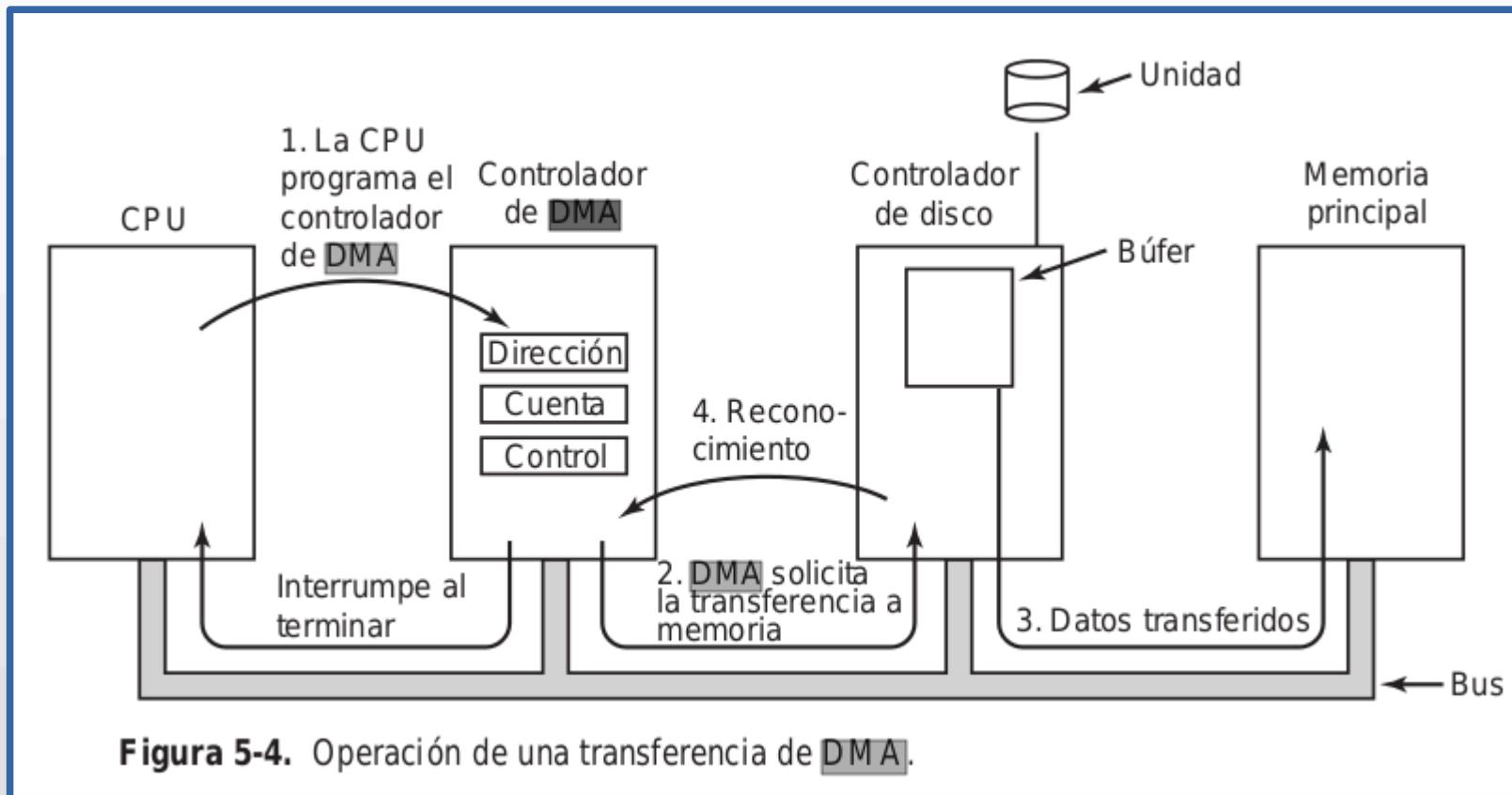


Figura 5-4. Operación de una transferencia de DMA.



Introducción a los Sistemas Operativos

Administración de
Archivos - I



- Versión: Noviembre 2017
- Palabras Claves: Archivo, Directorio, File System,

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Porque necesitamos archivos?

- Almacenar grandes cantidades de datos
- Tener almacenamiento a largo plazo
- Permitir a distintos procesos acceder al mismo conjunto de información



Archivo

- Entidad abstracta con nombre
- Espacio lógico continuo y direccionable
- Provee a los programas de datos
(entrada)
- Permite a los programas guardar datos
(salida)
- El programa mismo es información que
debe guardarse



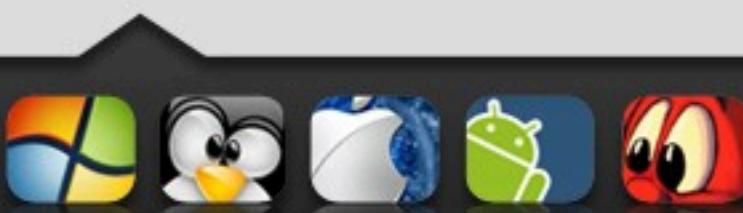
Archivos - Punto de vista del Usuario

- Que operaciones se pueden llevar a cabo
- Como nombrar a un archivo
- Como asegurar la protección
- Como compartir archivos
- No tratar con aspectos físicos
- Etc.



Archivos - Punto de vista del Diseño

- Implementar archivos
- Implementar directorios
- Manejo del espacio en disco
- Manejo del espacio libre
- Eficiencia y mantenimiento



Sistema de Manejo de Archivos

Conjunto de unidades de software que proveen los servicios necesarios para la utilización de archivos

- ✓ Crear
- ✓ Borrar
- ✓ Buscar
- ✓ Copiar
- ✓ Leer
- ✓ Escribir
- ✓ Etc.



Sistema de Manejo de Archivos (cont.)

- Facilita el acceso a los archivos por parte de las aplicaciones
- Permite la abstracción al programador, en cuanto al acceso de bajo nivel (el programador no desarrolla el soft de administración de archivos)



Objetivos del SO en cuanto a archivos

- Cumplir con la gestión de datos
- Cumplir con las solicitudes del usuario.
- Minimizar / eliminar la posibilidad de perder o destruir datos
 - ✓ Garantizar la integridad del contenido de los archivos
- Dar soporte de E/S a distintos dispositivos
- Brindar un conjunto de interfaces de E/S para tratamiento de archivos.



Tipos de Archivos

Archivos Regulares

Texto Plano

- ◆ Source File

Binarios

- ◆ Object File
- ◆ Executable File

Directorios

- ✓ Archivos que mantienen la estructura en el FileSystem



Atributos de un Archivo

- Nombre
- Identificador
- Tipo
- Localización
- Tamaño
- Protección, Seguridad y Monitoreo
 - ✓ Owner, Permisos, Password
 - ✓ Momento en que el usuario lo modifco, creo, accedio por ultima vez
 - ✓ ACLs



Ej: Tipos de archivos y atributos

		Referencias	Tamaño	Fecha hora de modificación	
# ls -la					
drwxr-xr-x	5	yoko yoko	4096	May 16 18:02	.
drwxr-xr-x	44	yoko yoko	4096	May 16 18:13	..
-rw-r--r--	1	yoko grupo1	0	May 16 18:01	archivo1
-rw-r--r--	1	yoko grupo1	0	May 16 17:54	archivo2
-rw-r--r--	1	yoko grupo1	0	May 16 17:57	archivo3
drwxr-xr-x	2	yoko grupo1	4096	May 16 17:55	directorio1
drwxr-xr-x	2	yoko grupo1	4096	May 16 17:54	directorio2
drwxr-xr-x	2	yoko grupo1	4096	May 16 18:02	directorio3
lrwxrwxrwx	1	yoko yoko	23	May 16 18:02	pepe99 -> directorio1/archivo9999

Diagrama que muestra la estructura de los resultados de la ejecución de 'ls -la'. Los resultados están organizados en columnas y se relacionan con los siguientes conceptos:

- Columna 1: Permisos (d : directorio, l : link (soft), - : archivo)
- Columna 2: Referencias
- Columna 3: Tamaño
- Columna 4: Fecha hora de modificación
- Columna 5: Nombre
- Columna 6: Usuario dueño y Grupo dueño



Directarios

- Contiene información acerca de archivos y directorios que están dentro de él
- El directorio es, en si mismo, un archivo
- Interviene en la resolución entre el nombre y el archivo mismo.
- Operaciones en directorios:
 - ✓ Buscar un archivo
 - ✓ Crear un archivo (entrada de directorio)
 - ✓ Borrar un archivo
 - ✓ Listar el contenido
 - ✓ Renombrar archivos
 - ✓ Etc.

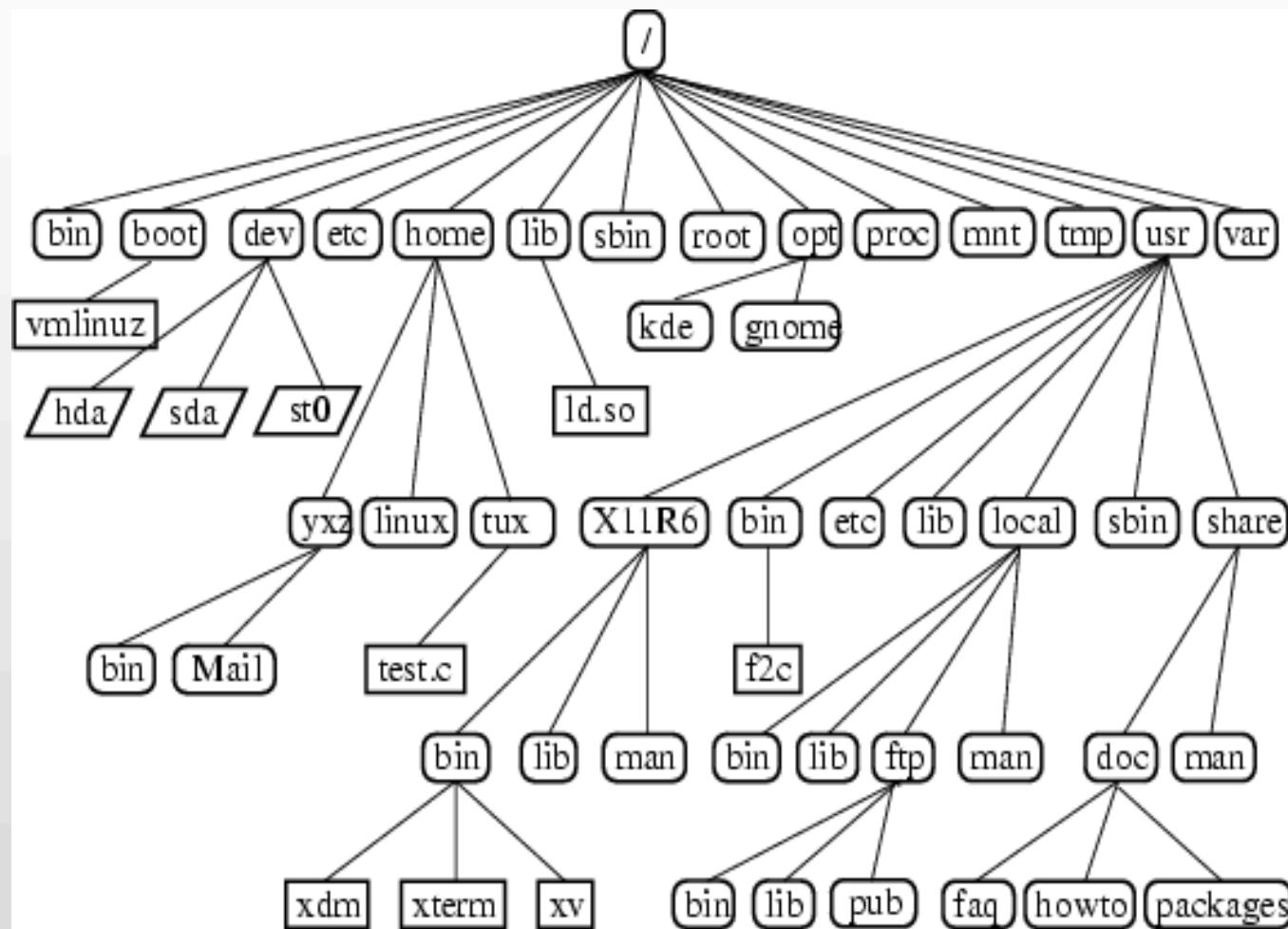


Directarios de Archivos (cont.)

- El uso de los directorios ayuda con:
 - ✓ La eficiencia: Localización rápida de archivos
 - ✓ Uso del mismo Nombre de archivo:
 - ◆ Diferentes usuarios pueden tener el mismo nombre de archivo
 - ✓ Agrupación: Agrupación lógica de archivos por propiedades/funciones:
 - Ejemplo: Programas Java, Juegos, Librerias, etc.



Estructura de Dir. Jerárquica o Arbol



Estructura de Directorios

- ✓ Los archivos pueden ubicarse siguiendo un path desde el directorio raíz y sus sucesivas referencias (**full pathname** del archivo o **PATH absoluto**)
- ✓ Distintos archivos pueden tener el mismo nombre pero el fullpathname es único



Estructura de Directorios

- ✓ El directorio actual se lo llama “directorio de trabajo (working directory)
- ✓ Dentro del directorio de trabajo, se pueden referenciar los archivos tanto por su **PATH absoluto** como por su **PATH relativo** indicando solamente la ruta al archivo desde el directorio de trabajo.



Identificación absoluta y relativa

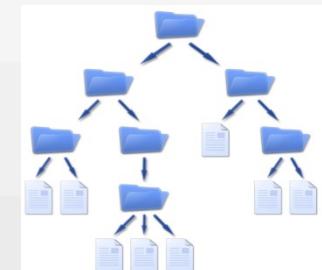
Tanto archivos como directorios se pueden identificar de manera:

- **Absoluta.** El nombre incluye todo el camino del archivo.

- </var/www/index.html>
- C:\windows\winhelp.exe

- **Relativa.** El nombre se calcula relativamente al directorio en el que se esté

- si estoy en el directorio </var/spool/mail/>
- Entonces es: [../../www/index.html](#)



Compartir archivos

- En un ambiente multiusuario se necesita que varios usuarios puedan compartir archivos
- Debe ser realizado bajo un esquema de protección:
 - ✓ Derechos de acceso
 - ✓ Manejo de accesos simultáneos



Protección

- El propietario/administrador debe ser capaz de controlar:
 - ✓ Que se puede hacer
 - ◆ Derechos de acceso
 - ✓ Quien lo puede hacer



Derechos de acceso

- Los directorios también tienen permisos, los cuales pueden permitir el acceso al mismo para que el usuario pueda usar el archivo siempre y cuando tenga permisos.



Derechos de acceso (cont.)

Execution

- ✓ El usuario puede ejecutar

Reading

- ✓ El usuario puede leer el archivo,

Appending

- ✓ El usuario puede agregar datos pero no modificar o borrar el contenido del archivo



Derechos de acceso (cont.)

Updating

- ✓ El usuario puede modificar, borrar y agregar datos. Incluye la creación de archivos, sobreescribirlo y remover datos

Changing protection

- ✓ El usuario puede modificar los derechos de acceso

Deletion

- ✓ El usuario puede borrar el archivo



Derechos de acceso

Owners (propietarios)

- ✓ Tiene todos los derechos
- ✓ Pueder dar derechos a otros usuarios. Se determinan clases:
 - ◆ Usuario específico
 - ◆ Grupos de usuarios
 - ◆ Todos (archivos públicos)



Ejemplo - Protección en UNIX

- ✓ Derechos de acceso son definidos independientemente para:

- ✓ (u) user - Owner (creator) of a file
- ✓ (g) group - Group
- ✓ (o) other - all other users of the UNIX system

✓ Derechos de Acceso:

- ✓ (r) **Read access right;** **List right for directory**
- ✓ (w) **Write access right;** **Includes delete/append rights**
- ✓ (x) **Execute access right;** **Traverse right for directories**

✓ Binary representation:

- ✓ (x): Bit 0 (+1)
- ✓ (w): Bit 1 (+2)
- ✓ (r): Bit 2 (+4)

✓ Rights can be combined

- ✓ Read+Write access right: 6
- ✓ Read+Execute access right: 3
- ✓ Read-only: 2

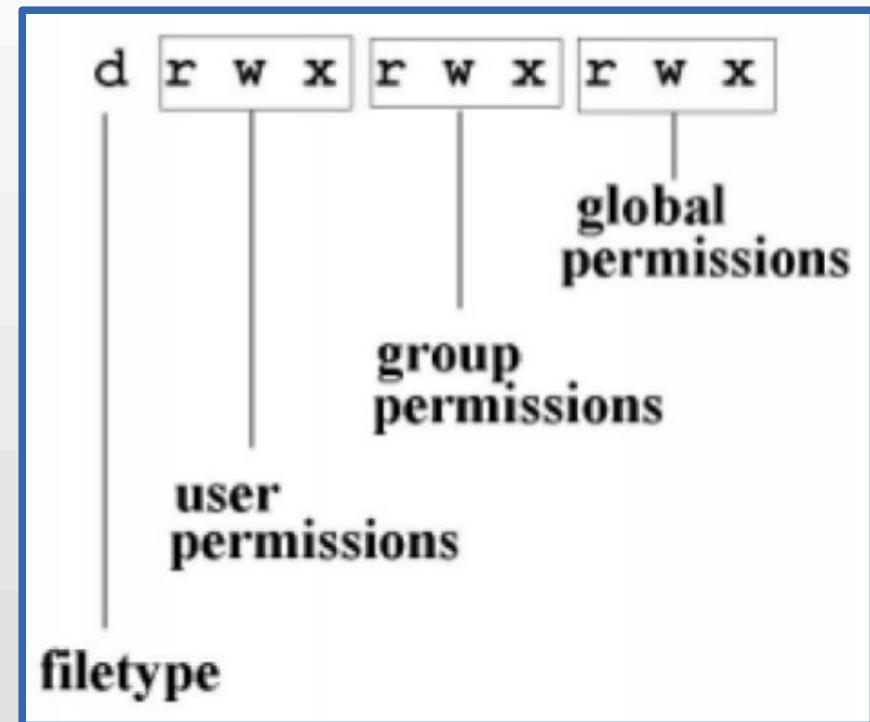


Ejemplo - Protección en UNIX

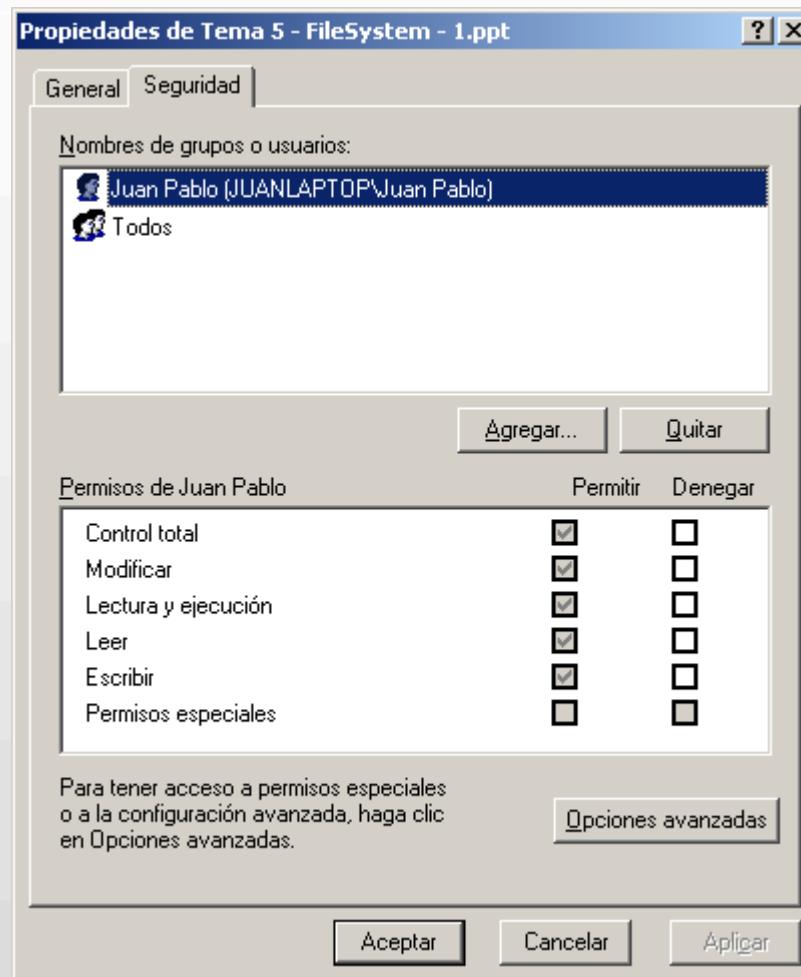
Los permisos que se pueden dar o quitar son:

- r - de lectura
- w - de escritura
- x - de ejecución

```
$ ls -l
drwxrwxr-x 4 www      www      ..
-rw-rw-r-- 1 www      www      x_windows.tex
lrwxrwxrwx 1 lee      lee      img -> ../linux/img/
-rw-rw-r-- 1 lee      lee      test.log
```



Ejemplo - Protección en Windows



Introducción a los Sistemas Operativos

Administración de
Archivos - II



Versión: Noviembre 2017

Palabras Claves: Archivo, Directorio, File System, Asignación, Espacio Libre

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Metas del Sistema de Archivos

- Brindar espacio en disco a los archivos de usuario y del sistema.
- Mantener un registro del espacio libre. Cantidad y ubicación del mismo dentro del disco.



Conceptos

Sector

- ✓ Unidad de almacenamiento utilizada en los Discos Rígidos

Bloque/Cluster

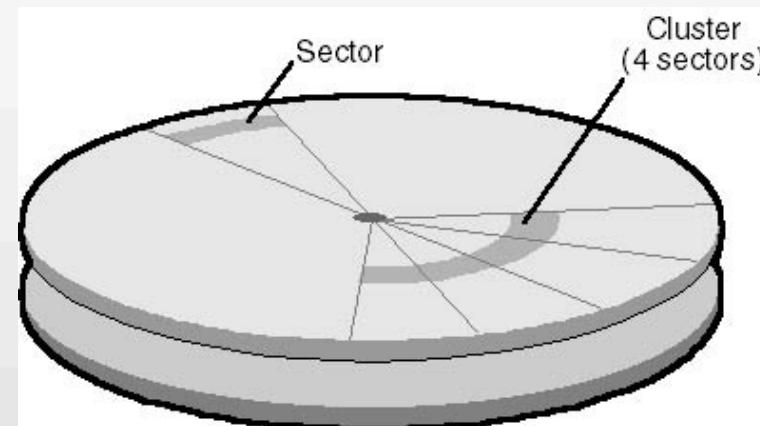
- ✓ Conjuntos de sectores consecutivos

File System

- ✓ Define la forma en que los datos son almacenados

FAT: File Allocation Table

- ✓ Contiene información sobre en que lugar están alojados los distintos archivos



Pre-asignación

- Se necesita saber cuánto espacio va a ocupar el archivo en el momento de su creación
- Se tiende a definir espacios mucho más grandes que lo necesario
- Posibilidad de utilizar sectores contiguos para almacenar los datos de un archivo
- ¿Qué pasa cuando el archivo supera el espacio asignado?

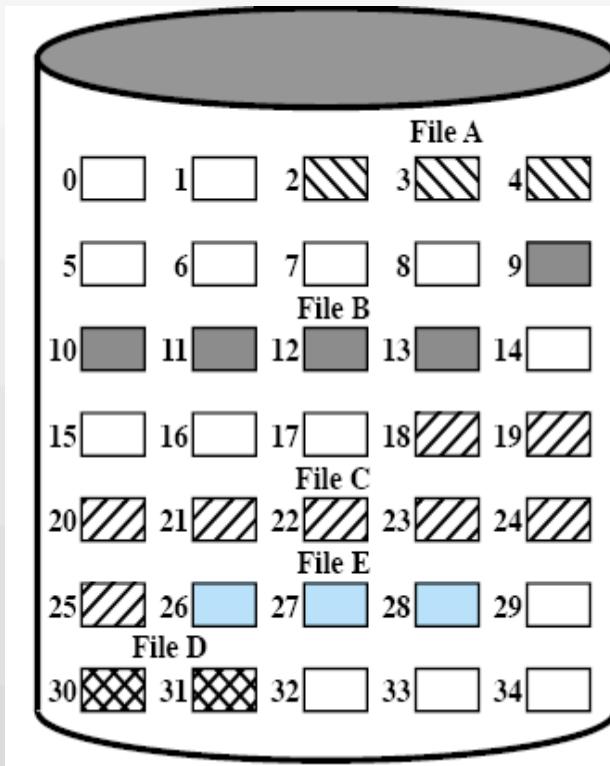


Asignación Dinámica

- El espacio se solicita a medida que se necesita
- Los bloques de datos pueden quedar de manera no contigua



Formas de Asignación - Continua



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Que sucedería si necesitamos agregar un nuevo archivo de 6 bloques?



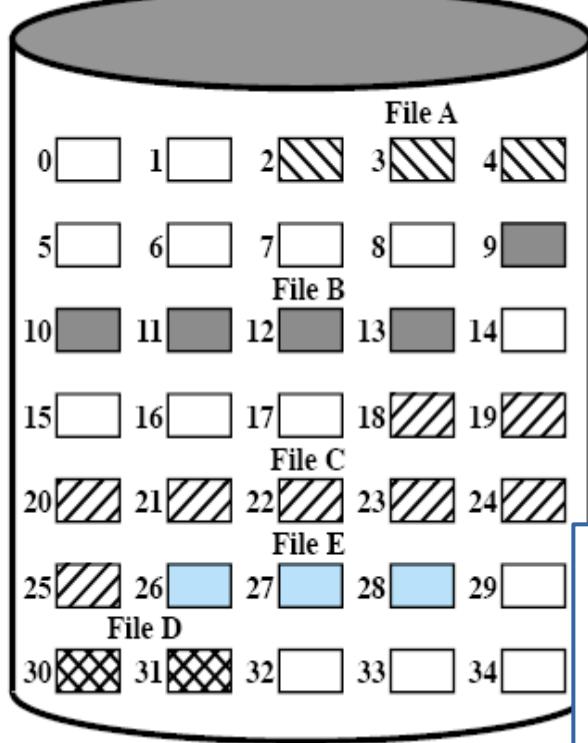
Formas de Asignación - Continua

- Conjunto continuo de bloques son utilizados
- Se requiere una pre-asignación
 - ✓ Se debe conocer el tamaño del archivo durante su creación
- File Allocation Table (FAT) es simple
 - ✓ Sólo una entrada que incluye Bloque de inicio y longitud
- El archivo puede ser leído con una única operación
- Puede existir fragmentación externa
 - ✓ Compactación

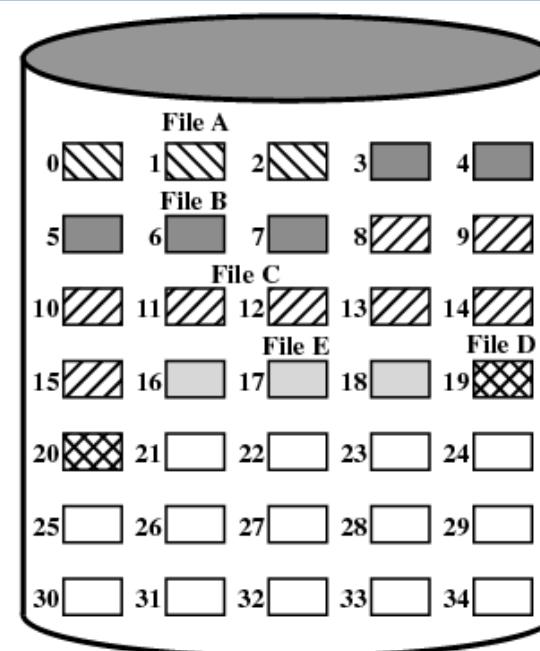
File Allocation Table		
File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



Compactación



File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3



Figure 12.8 Contiguous File Allocation (After Compaction)

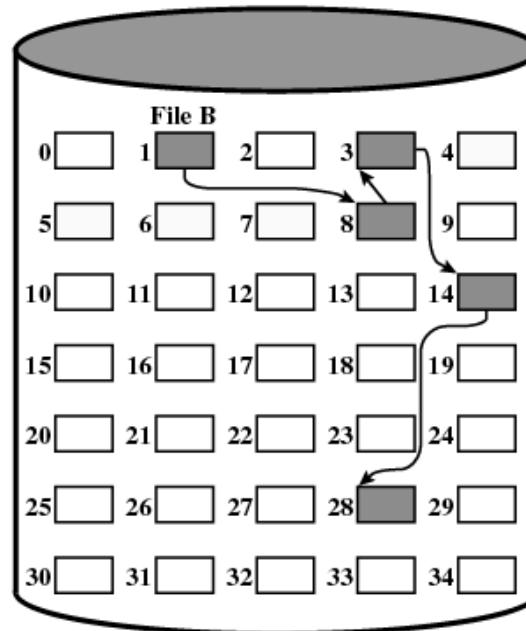
Formas de Asignación - Continua

Problemas de la técnica

- ✓ Encontrar bloques libres continuos en el disco
- ✓ Incremento del tamaño de un archivo



Formas de Asignación - Encadenada



File Allocation Table		
File Name	Start Block	Length
...
File B	1	5
...

Figure 12.9 Chained Allocation



Formas de Asignación - Encadenada

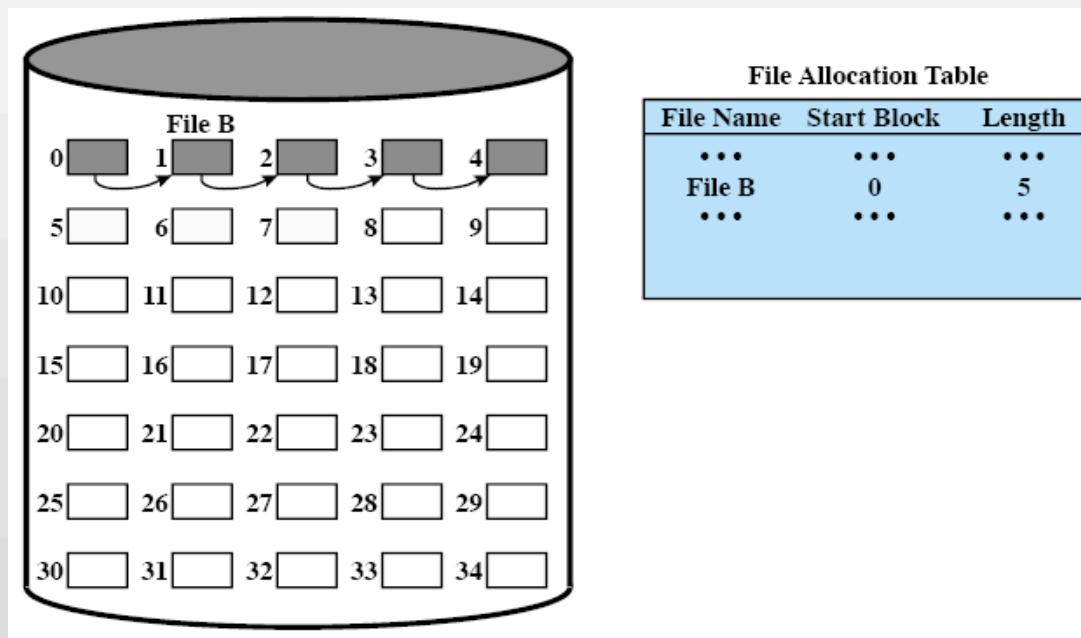
- Asignación en base a bloques individuales
- Cada bloque tiene un puntero al próximo bloque del archivo
- File allocation table
 - ✓ Única entrada por archivo: Bloque de inicio y tamaño del archivo
- No hay fragmentación externa
- Útil para acceso secuencial (no random)
- Los archivos pueden crecer bajo demanda
- No se requieren bloques contiguos

File Allocation Table		
File Name	Start Block	Length
...
File B	1	5
...



Formas de Asignación - Encadenada

- ✓ Se pueden consolidar los bloques de un mismo archivo para garantizar cercanía de los bloques de un mismo archivo.



Formas de Asignación - Indexada

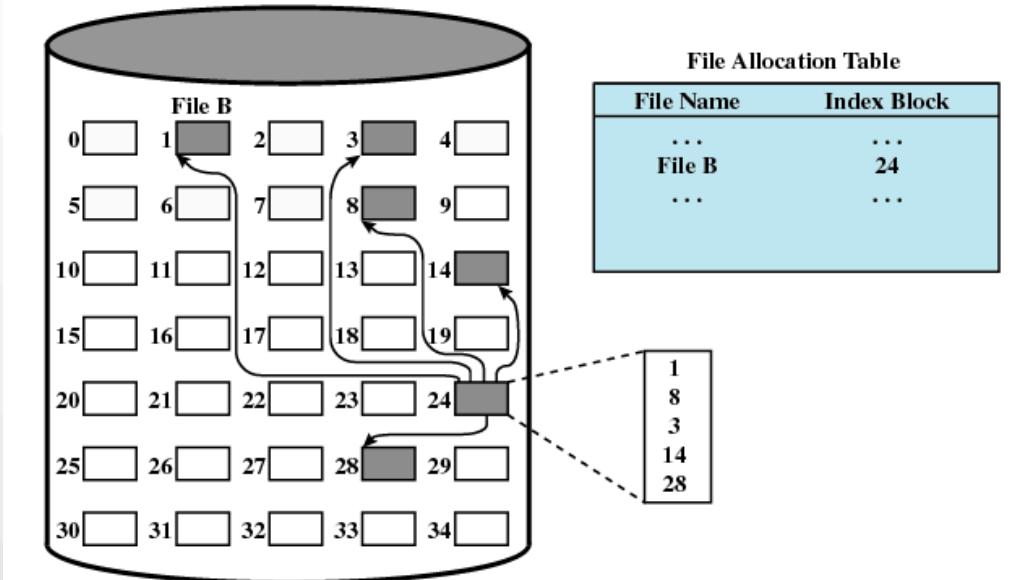


Figure 12.11 Indexed Allocation with Block Portions

- La FAT contiene un puntero al bloque índice
- El bloque índice no contiene datos propios del archivo, sino que contiene un índice a los bloques que lo componen



Formas de Asignación - Indexada

- Asignación en base a bloques individuales
- No se produce Fragmentación Externa
- El acceso “random” a un archivo es eficiente
- File Allocation Table
 - ✓ Única entrada con la dirección del bloque de índices (index node / i-node)

File Allocation Table	
File Name	Index Block
...	...
File B	24
...	...



Formas de Asignación - Indexada

- Variante: asignación por secciones
- A cada entrada del bloque índice se agrega el campo longitud
- El índice apunta al primer bloque de un conjunto almacenado de manera contigua

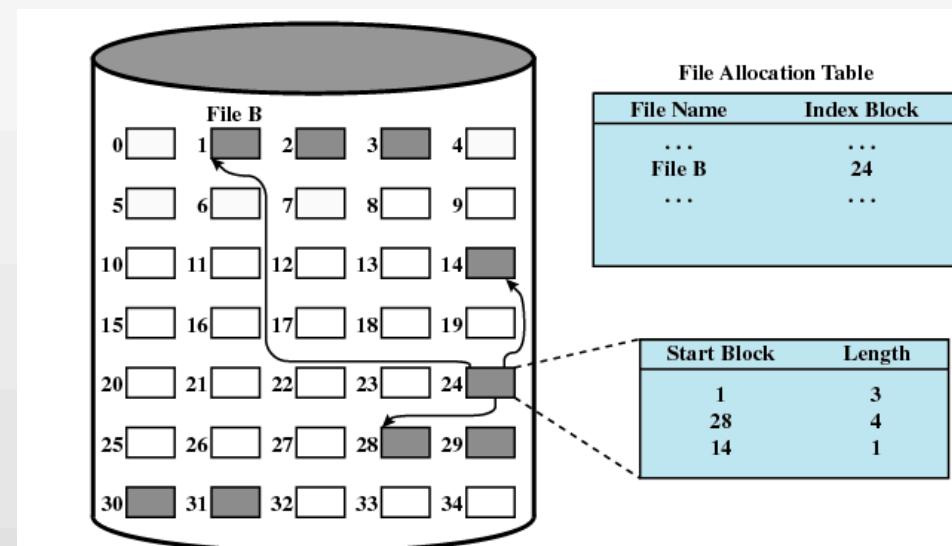


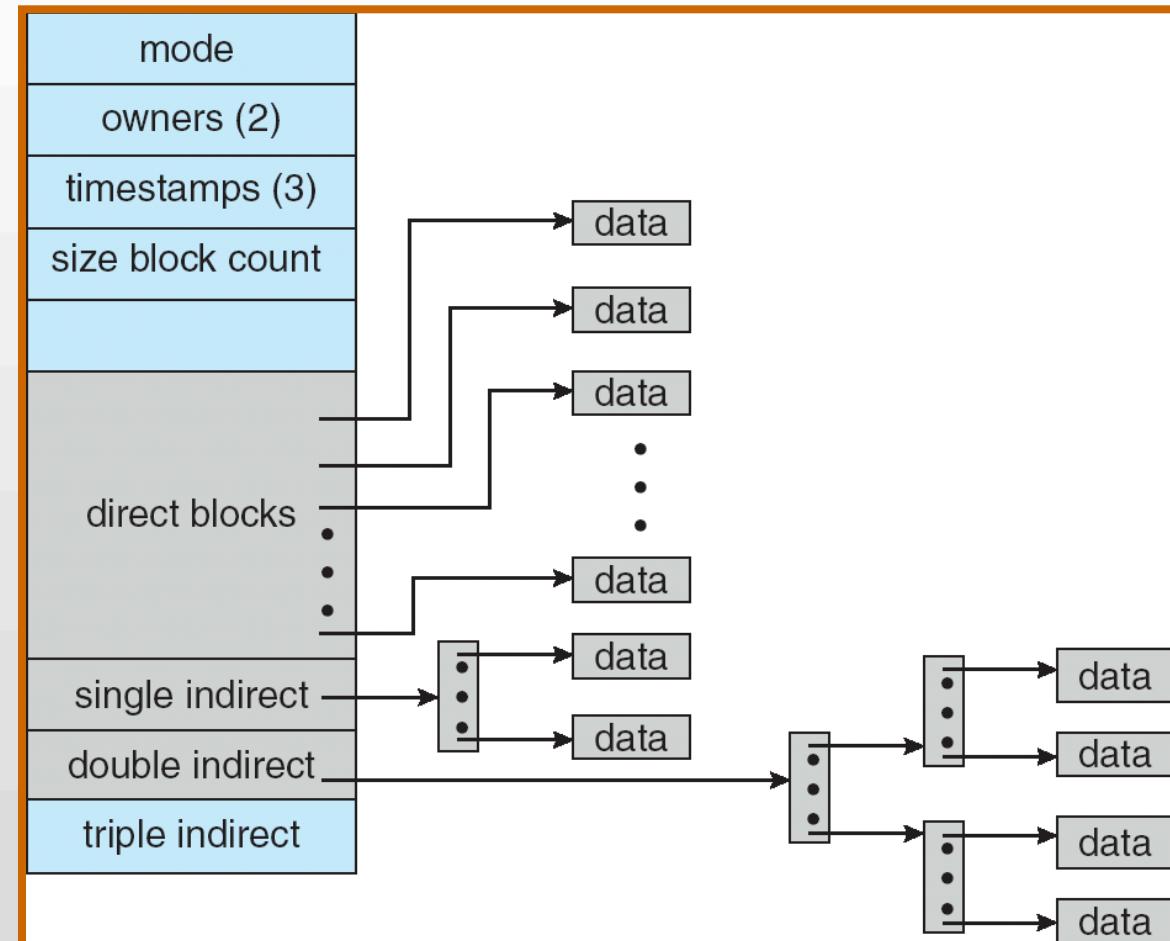
Figure 12.12 Indexed Allocation with Variable-Length Portions



Formas de Asignación - Indexada

Variante: niveles de indirección

- Existen bloques directos de datos
- Otros bloques son considerados como bloque índices (apuntan a varios bloques de datos)
- Puede haber varios niveles de indirección



Asignación Indexada - Ejemplo

Cada I-NODO contiene 9 direcciones a los bloques de datos, organizadas de la siguiente manera:

- ◆ 7 de direccionamiento directo.
- ◆ 1 de direccionamiento indirecto simple
- ◆ 1 de direccionamiento indirecto doble

Si cada bloque es de 1KB y cada dirección usada para referenciar un bloque es de 32 bits:

✓ ¿Cuántas referencias (direcciones) a bloque pueden contener un bloque de disco?

$$1 \text{ KB} / 32 \text{ bits} = 256 \text{ direcciones}$$

✓ ¿Cuál sería el tamaño máximo de un archivo?

$$(7 + 256 + 256^2) * 1 \text{ KB} = 65799 \text{ KB} = 64,25 \text{ MB}$$



Gestión de Espacio Libre

Control sobre cuáles de los bloques de disco están disponibles.

Alternativas

- *Tablas de bits*
- *Bloques libres encadenados*
- *Indexación*



Espacio Libre - Tabla de bits

- Tabla (vector) con 1 bit por cada bloque de disco
- Cada entrada:
 - ✓ 0 = bloque libre 1 = bloque en uso
- Ventaja
 - ✓ Fácil encontrar un bloque o grupo de bloques libres.
- Desventaja
 - ✓ Tamaño del vector en memoria
tamaño disco bytes / tamaño bloque en sistema archivo
Eje: Disco 16 Gb con bloques de 512 bytes → 32 Mb.



Espacio Libre - Tabla de bits (cont.)

Ejemplo

00111

00001

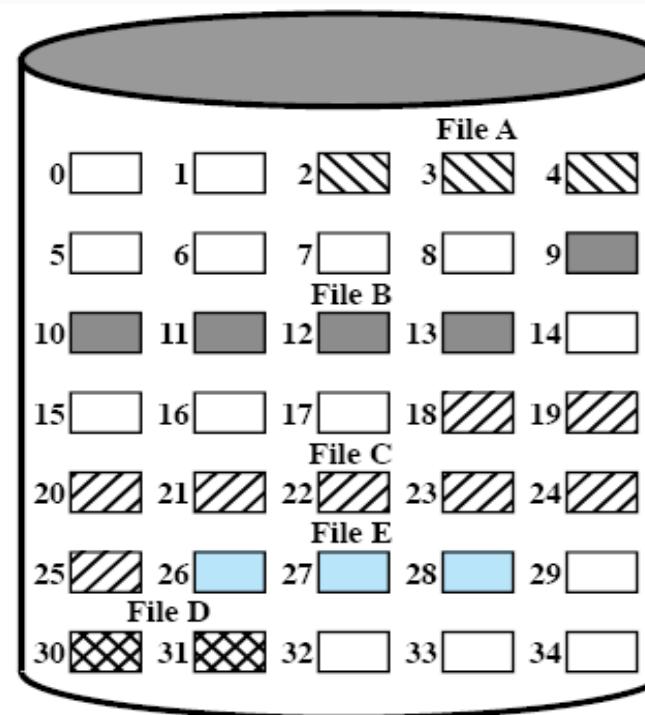
11110

00011

11111

11110

11000

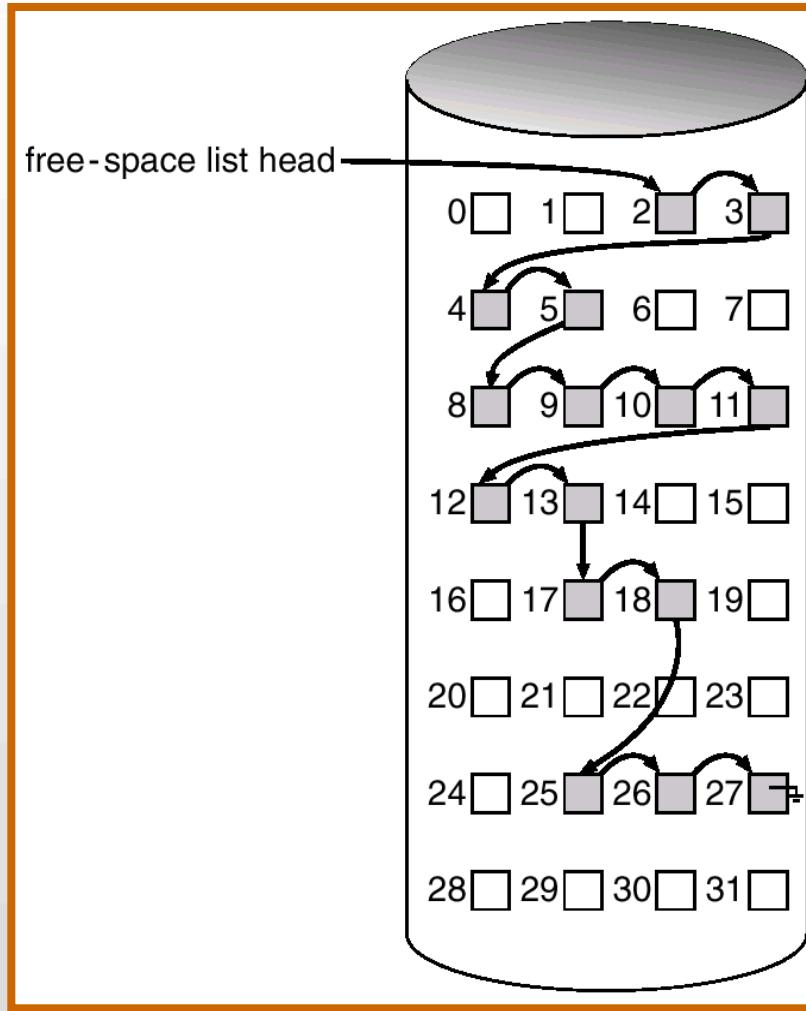


Espacio Libre - Bloques Encadenados

- Se tiene un puntero al primer bloque libre.
- Cada bloque libre tiene un puntero al siguiente bloque libre
- Ineficiente para la búsqueda de bloques libres → Hay que realizar varias operaciones de E/S para obtener un grupo libre.
- Problemas con la pérdida de un enlace
- Difícil encontrar bloques libres consecutivos



Espacio Libre - Bloques Encadenados



Espacio Libre - Indexación (o agrupamiento)

- Variante de “bloques libres encadenados”
- El primer bloque libre contiene las direcciones de N bloques libres.
- Las N-1 primeras direcciones son bloques libres.
- La N-ésima dirección referencia otro bloque con N direcciones de bloques libres.



Introducción a los Sistemas Operativos

Administración de
Archivos - III



Versión: Noviembre 2017

Palabras Claves: Archivo, File System, Directorio, UNIX, I-NODO, Windows, FAT

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



UNIX - Manejo de archivos

Tipos de Archivos

- ✓ Archivo común
- ✓ Directorio
- ✓ Archivos especiales (dispositivos /dev/sda)
- ✓ Named pipes (comunicación entre procesos)
- ✓ Links (comparten el i-nodo, solo dentro del mismo filesystem)
- ✓ Links simbólicos (tiene i-nodo propio, para filesystems diferentes)

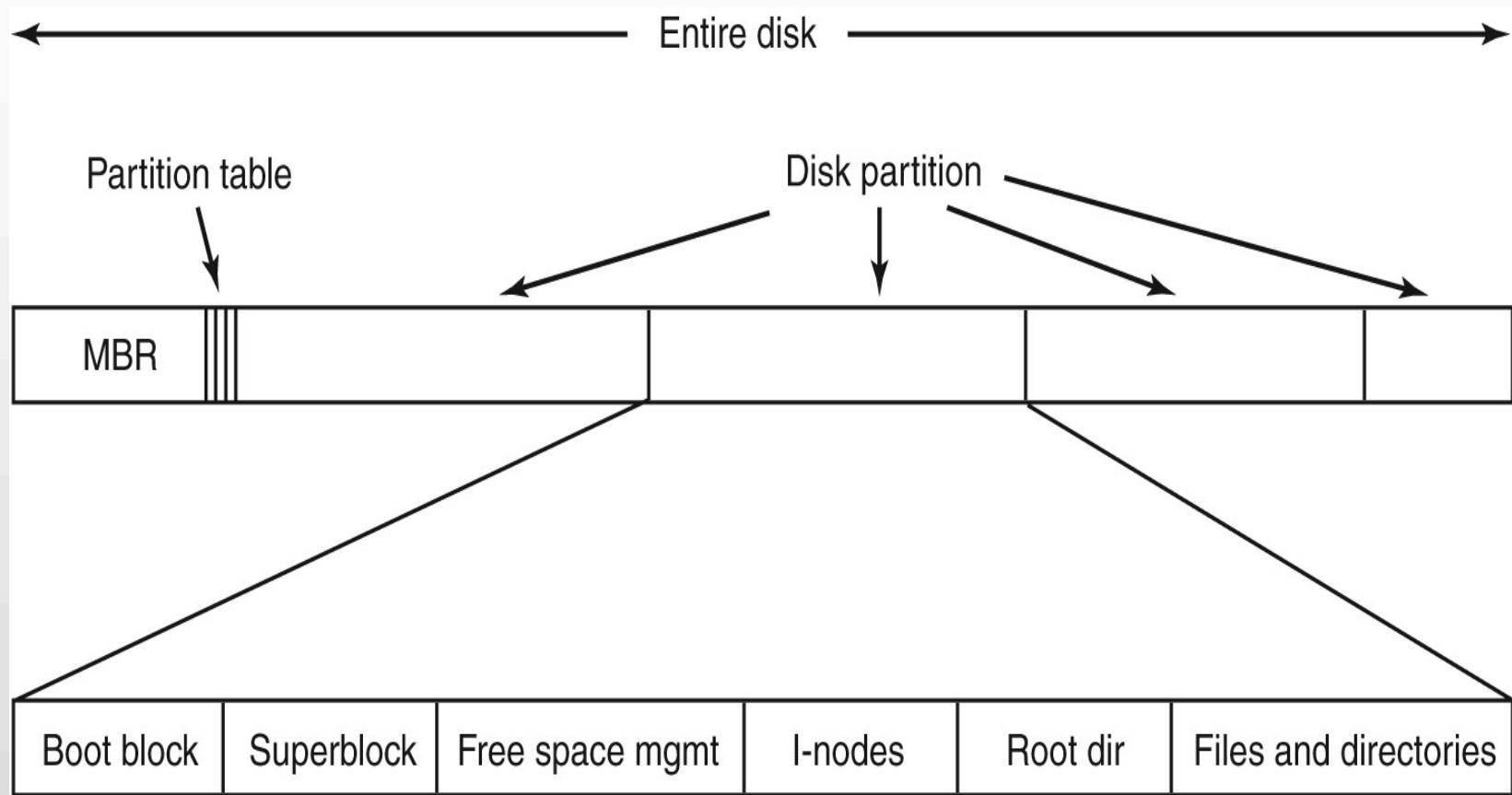


UNIX - Estructura del Volumen

- Cada disco físico puede ser dividido en uno o mas volúmenes. Cada volumen o partición contiene un sistema de archivos. Cada sistema de archivos contiene:
 - Boot Block: Código para bootear el S.O.
 - Superblock: Atributos sobre el File System
 - Bloques/Clusters libres
 - I-NODE Table: Tabla que contiene todos los I-NODOS
 - I-NODO: Estructura de control que contiene la información clave de un archivo
 - Data Blocks: Bloques de datos de los archivos



UNIX - Estructura del Volumen



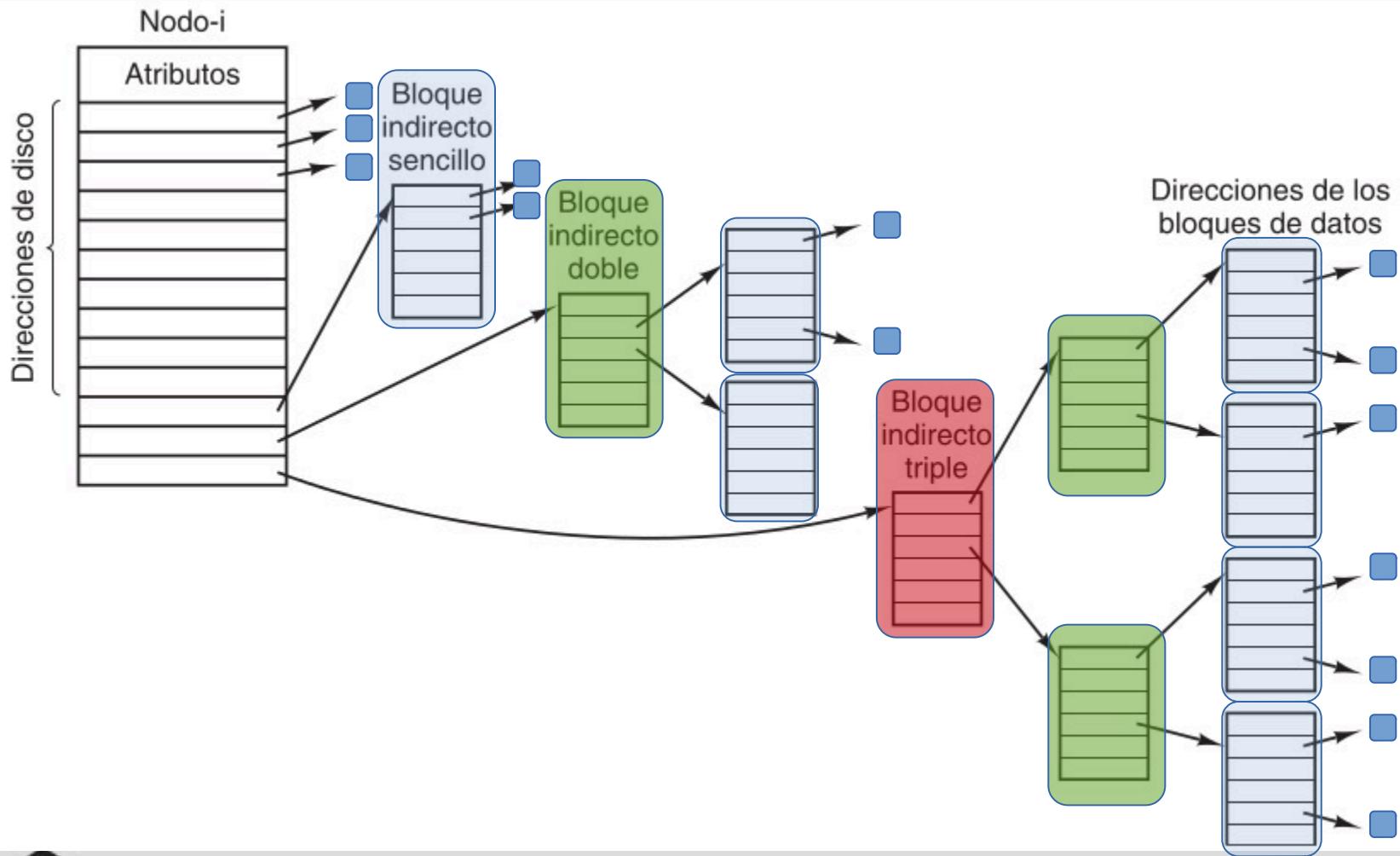
UNIX - Información del i-nodo

Table 12.4 Information in a UNIX Disk-Resident Inode

File Mode	16-bit flag that stores access and execution permissions associated with the file. 12-14 File type (regular, directory, character or block special, FIFO pipe 9-11 Execution flags 8 Owner read permission 7 Owner write permission 6 Owner execute permission 5 Group read permission 4 Group write permission 3 Group execute permission 2 Other read permission 1 Other write permission 0 Other execute permission
Link Count	Number of directory references to this inode
Owner ID	Individual owner of file
Group ID	Group owner associated with this file
File Size	Number of bytes in file
File Addresses	39 bytes of address information
Last Accessed	Time of last file access
Last Modified	Time of last file modification
Inode Modified	Time of last inode modification



UNIX - I-NODO



UNIX - Directorios (cont)

Buscar el i-nodo del archivo /usr/ast/mbox

Root directory		I-node 6 is for /usr	Block 132 is /usr directory	I-node 26 is for /usr/ast	Block 406 is /usr/ast directory
1	.		6 •	Mode	26 •
1	..		1 ..	size	6 ..
4	bin		19 dick	times	64 grants
7	dev	132	30 erik	406	92 books
14	lib		51 jim		60 mbox
9	etc		26 ast		81 minix
6	usr		45 bal		17 src
8	tmp				

Looking up usr yields i-node 6

I-node 6 says that /usr is in block 132

/usr/ast is i-node 26

I-node 26 says that /usr/ast is in block 406

/usr/ast/mbox is i-node 60



Windows - File Systems Soportados

- CD-ROM File System (CDFS) → CD
- Universal Disk Format (UDF) → DVD, Blu-Ray
- File Allocation Table
 - FAT12 → MS-DOS v3.3 a 4.0 (año 1980), floppy
 - FAT16 → MS-DOS 6.22, nombres cortos de archivo
 - FAT32 → MS-DOS 7.10, nombres largos pero no soportados en MS-DOS
- New Technology File System (NTFS)



Windows - FAT

- FAT (File Allocation Table) es un sistema de archivos utilizado originalmente por DOS y Windows 9x
- ¿Porqué Windows aun soporta FAT file systems?:
 - ✓ Por compatibilidad con otro SO en sistemas multiboot
 - ✓ Para permitir upgrades desde versiones anteriores
 - ✓ Para formato de dispositivos como diskettes
- Las distintas versiones de FAT se diferencian por un número que indica la cantidad de bits que se usan para identificar diferentes bloques o clusters:
 - FAT12
 - FAT16
 - FAT32



Windows - FAT

- Se utiliza un mapa de bloques del sistema de archivos, llamado FAT.
- La FAT tiene tantas entradas como bloques.
- La FAT, su duplicado y el directorio raíz se almacenan en los primeros sectores de la partición



FAT format organization



Windows - FAT

- Se utiliza un esquema de ASIGNACION ENCADENADA.
- La única diferencia es que el puntero al proximo bloque está en la FAT y no en los bloques
- Bloques libres y dañados tienen codigos especiales

DIRECTORIO		1er bloque	Tamaño
Nombre			
FICH_A		7	4
FICH_B		4	1
FICH_C		2	3

FAT	
Tamaño del disco	0
6	1
14	2
EOF	3
EOF	4
5	5
5	6
3	7
EOF	8
LIBRE	9
LIBRE	10
LIBRE	11
LIBRE	12
DAÑADO	13
8	14
LIBRE	15
...	...



Windows - FAT12

- ✓ En sistemas FAT12, al utilizarse 12 bits para la identificación del sector, la misma se limita a 2^{12} (4096) sectores
 - ✓ Windows utiliza tamaños de sector desde los 512 bytes hasta 8 KB, lo que limita a un tamaño total de volume de 32 MB → $2^{12} * 8 \text{ KB}$
 - ✓ Windows utiliza FAT12 como Sistema de archivos de los disketts de 3,5 y 12 pulgadas que pueden almacenar hasta 1,44 MB de datos



Windows - FAT16

- ✓ FAT16 al utilizar 16 bits para identificar cada sector puede tener hasta 2^{16} (65.536) sectores en un volúmen
 - ✓ En windows el tamaño de sector en FAT16 varía desde los 512 bytes hasta los 64 KB, lo que limita a un tamaño máximo de volume de 4 GB.
 - ✓ El tamaño de sector dependía del tamaño del volume al formatearlo



Windows - FAT32

- FAT32 fue el Filesystem mas reciente de la línea (posteriormente salió exFAT que algunos lo conocen como FAT64)
- FAT32 utiliza 32 bits para la identificación de sectores, pero reserva los 4 bits superiores, con lo cual efectivamente solo se utilizan 28 bits para la identificación:
 - ✓ El tamaño de sector en FAT 32 puede ser de hasta 32 KB, con lo cual tiene una capacidad teórica de direccionar volúmenes de hasta 8 TB
 - ✓ El modo de identificación y acceso de los sectores lo hace mas eficiente que FAT16. Con tamaño de sector de 512 bytes, puede direccionar volúmenes de hasta 128 GB.



Windows - FAT

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB



Windows - NTFS

- NTFS es el filesystem nativo de Windows desde Windows NT
- NTFS usa 64-bit para referenciar sectores
 - ✓ Teoricamente permite tener volúmenes de hasta 16 Exabytes (16 billones de GB)
- ¿Porqué usar NTFS en lugar de FAT? FAT es simple, mas rápido para ciertas operaciones, pero NTFS soporta:
 - ✓ Tamaños de archivo y de discos mayores
 - ✓ Mejora performance en discos grandes
 - ✓ Nombres de archivos de hasta 255 caracteres
 - ✓ Atributos de seguridad
 - ✓ Transaccional

