

# *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.

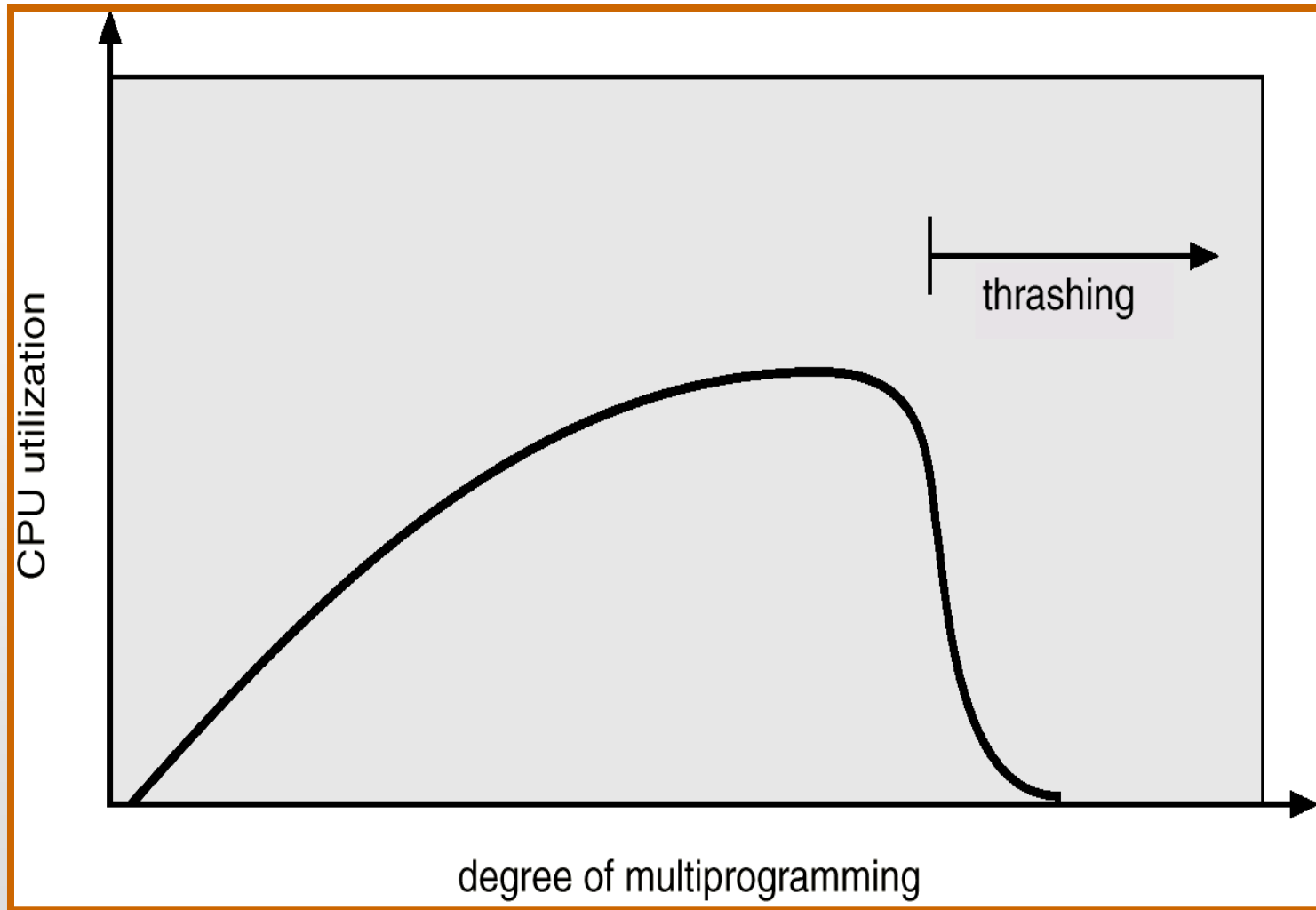


# *Ciclo del thrashing*

- 1) El kernel monitorea el uso de la CPU.
- 2) Si hay baja utilización  $\Rightarrow$  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)



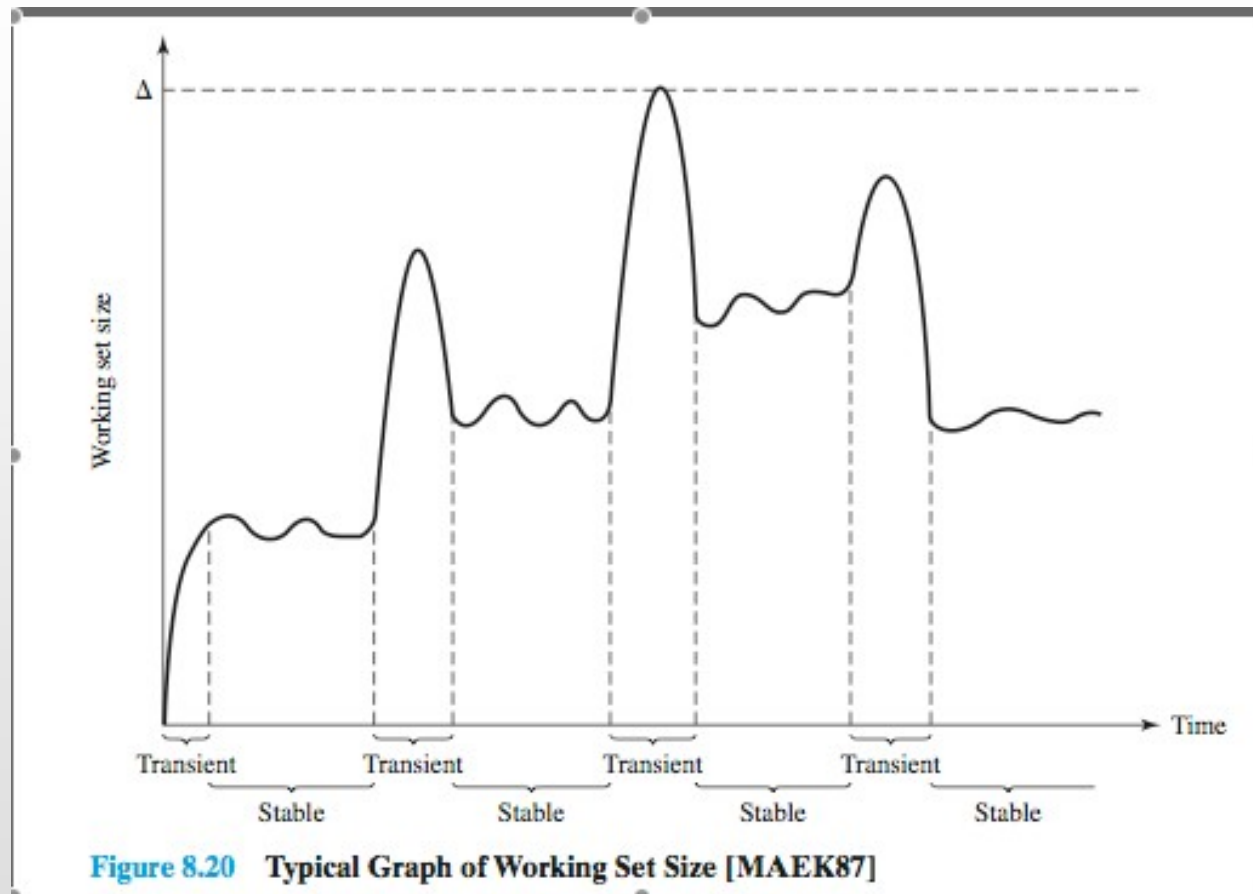


# *El modelo de localidad*

- ✓ 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



# *El modelo de localidad (cont.)*

- ✓ Un programa se compone de varias localidades.
- ✓ Ejemplo: Cada rutina será una nueva localidad: se referencian sus direcciones (cercanas) 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 ( $\Delta$ ): las referencias de memoria más recientes.
- ✓ Working set: es el conjunto de páginas que tienen las más recientes  $\Delta$  referencias a páginas.

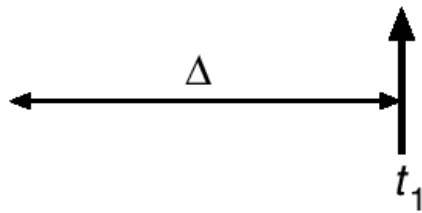


# Ejemplo de working set

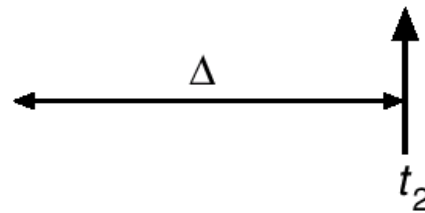
✓  $\Delta=10$

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 ...



$$WS(t_1) = \{1, 2, 5, 6, 7\}$$



$$WS(t_2) = \{3, 4\}$$



# La selección del $\Delta$

- ✓  $\Delta$  chico: no cubrirá la localidad
- ✓  $\Delta$  grande: puede tomar varias localidades



# Medida del working set

- ✓  $m$  = cantidad frames disponibles
- ✓  $WSS_i$  = medida del working set del proceso  $p_i$ .
- ✓  $\sum WSS_i = D$ ;
- ✓  $D$  = 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

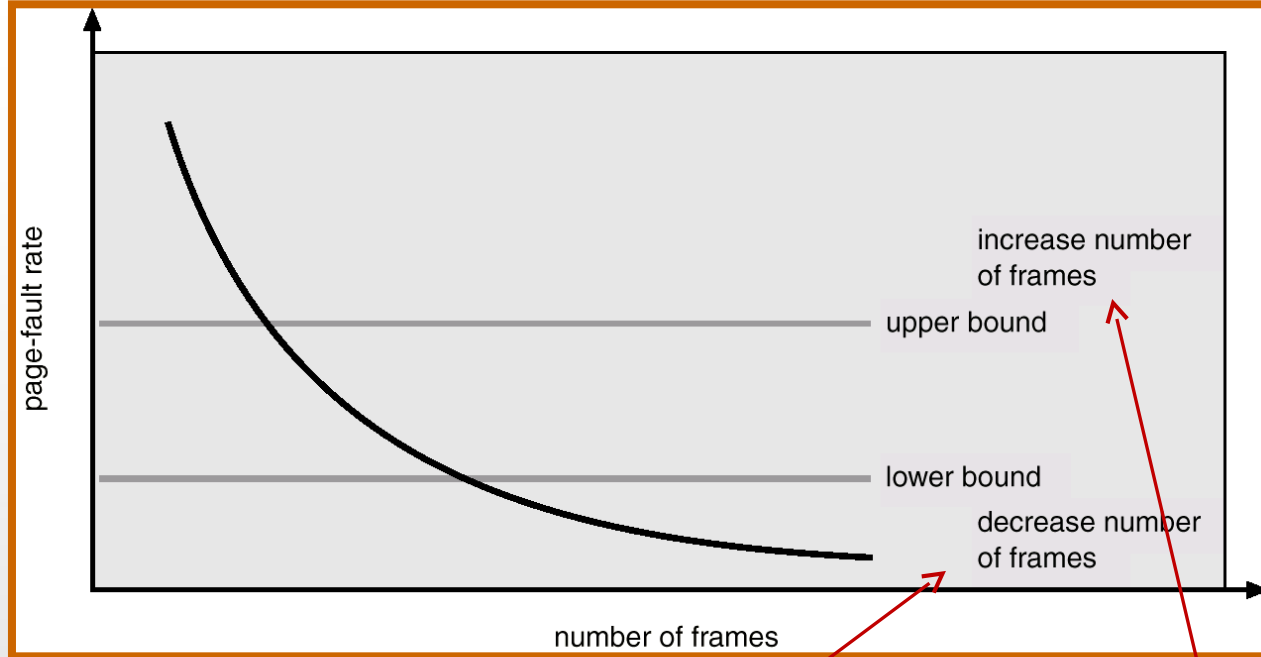


# Prevención del thrashing por PFF

- ✓ La técnica PFF (Page Fault Frequency 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  $\Rightarrow$  Se necesitan más frames
- ✓ PFF baja  $\Rightarrow$  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.  $\Rightarrow$  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  $\Rightarrow$  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



# 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**

✓ Programa 2:

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

**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**”



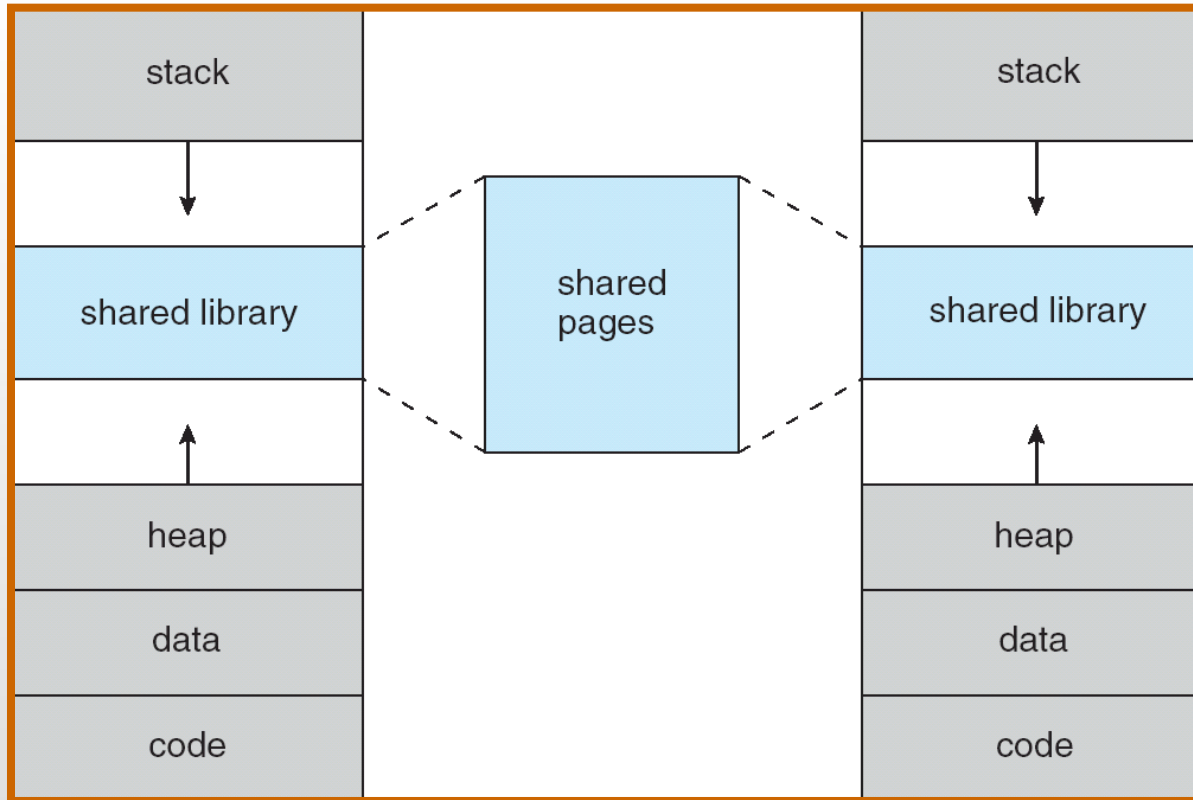
# Memoria Compartida

- ✓ 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

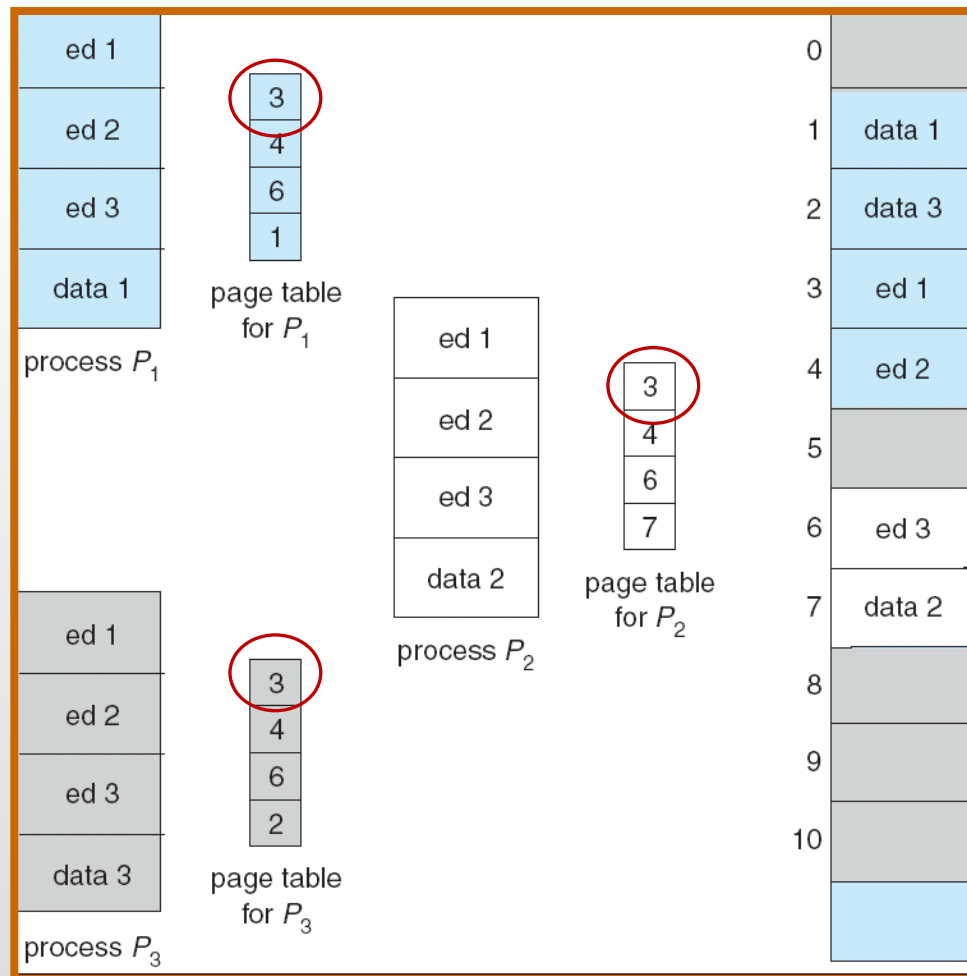




# Memoria Compartida (cont.)



# Memoria Compartida (cont.)

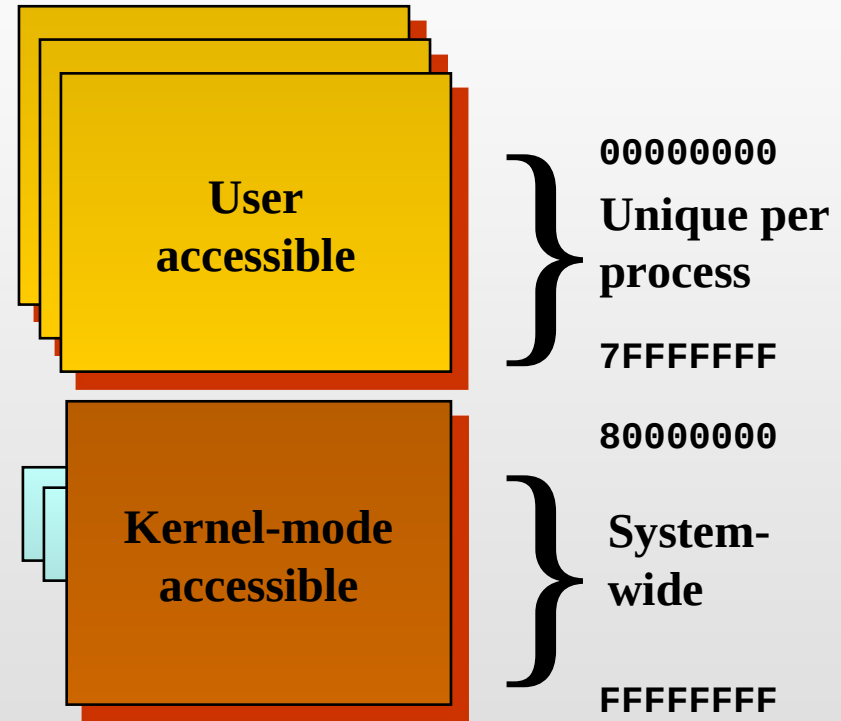


# Memoria Compartida - Ej. Windows

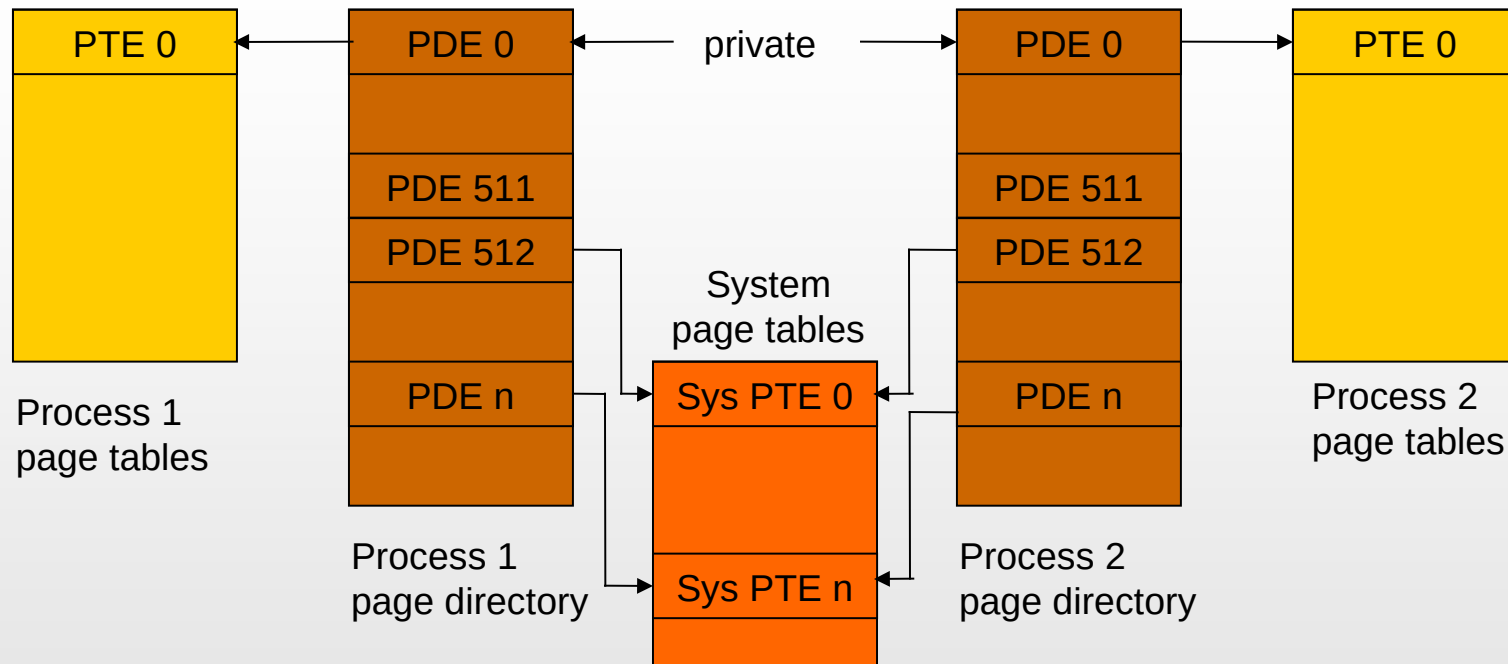
## ✓ Process space

contains:

- ✓ The application you're running (.EXE and .DLLs)
- ✓ All static storage defined by the application



# 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



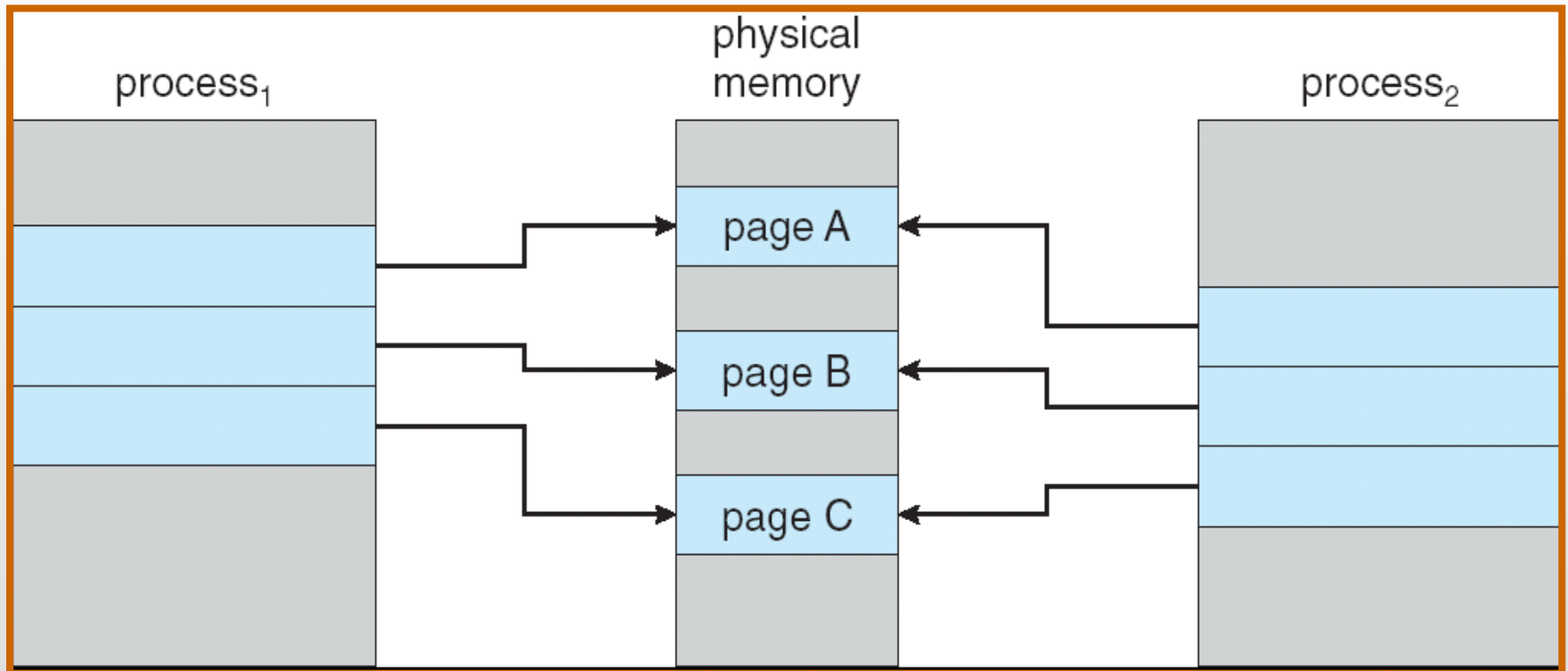
# Copia en Escritura

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



# Copia en Escritura (cont.)

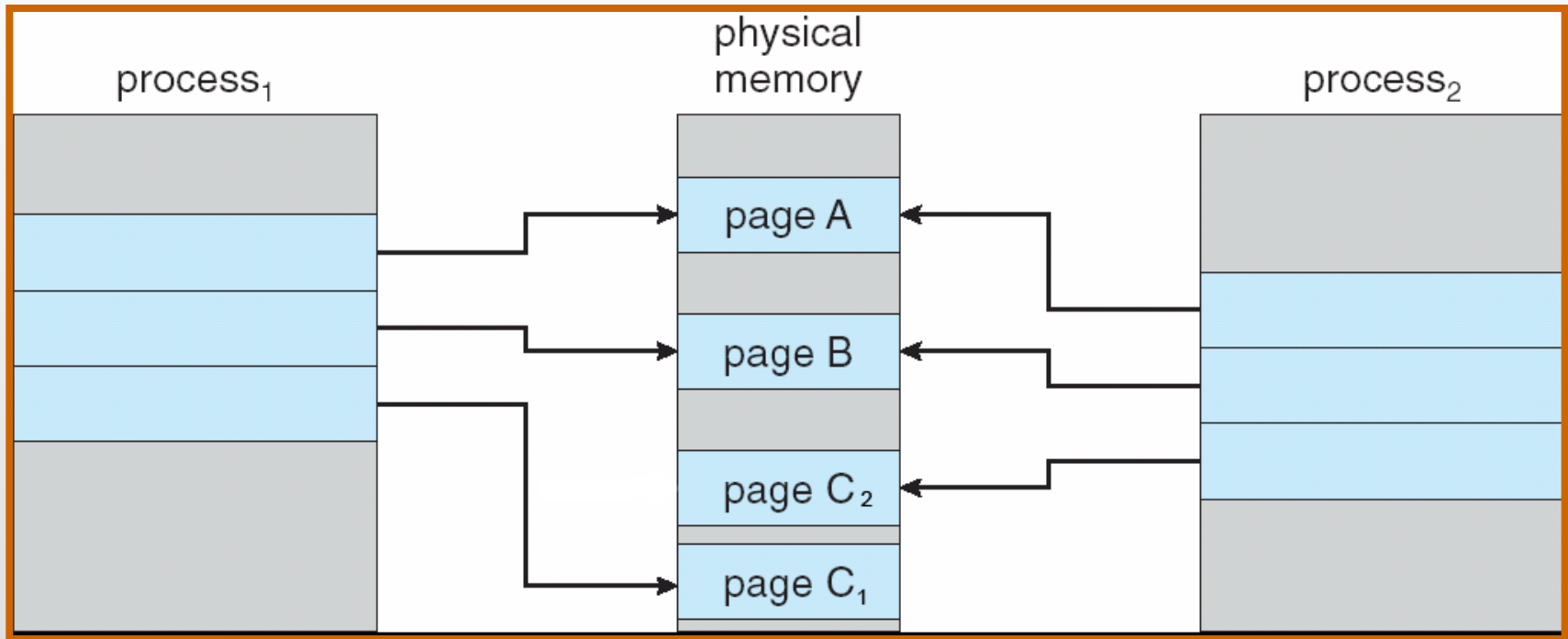
El Proceso 1 Modifica la Página C (Antes)





# Copia en Escritura (cont.)

El Proceso 1 Modifica la Página C (Despues)



# Área de Intercambio

## ☑ Sobre el Área utilizada

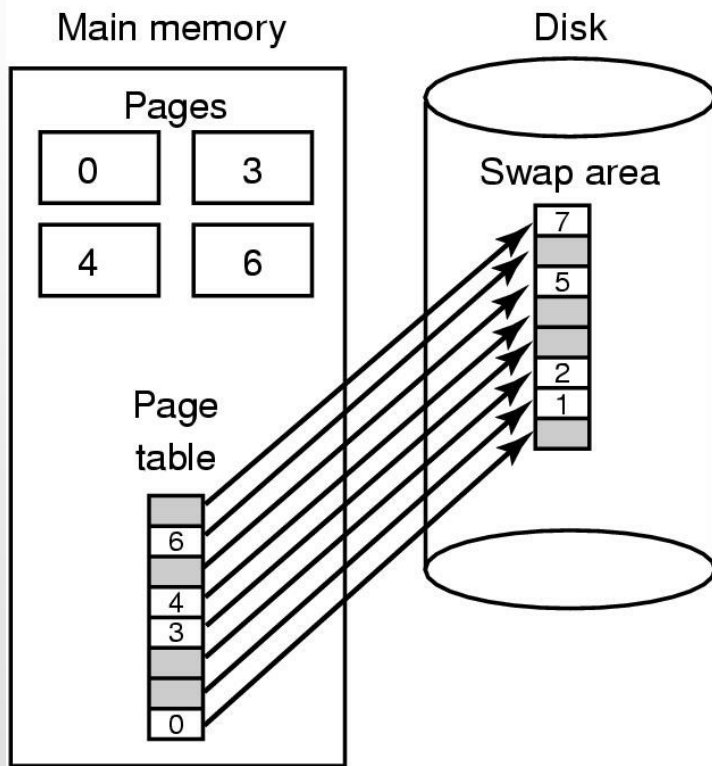
- ✓ Área dedicada, separada del Sistema de Archivos (Por ejemplo, en Linux)
- ✓ Un archivo dentro del Sistema de Archivos (Por ejemplo, Windows)

## ☑ Técnicas para la Administración:

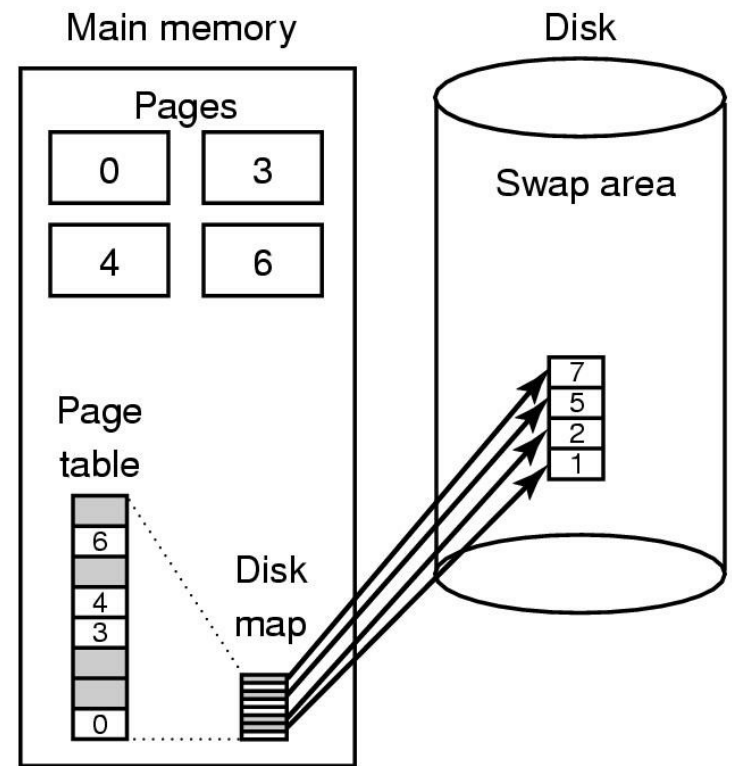
- a) Cada vez que se crea un proceso **se reserva una zona del área de intercambio** igual al tamaño de imagen del proceso. A cada proceso se le asigna la dirección en disco de su área de intercambio. La lectura se realiza sumando el número de página virtual a la dirección de comienzo del área asignada al proceso.
- b) **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 disco, como podemos 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

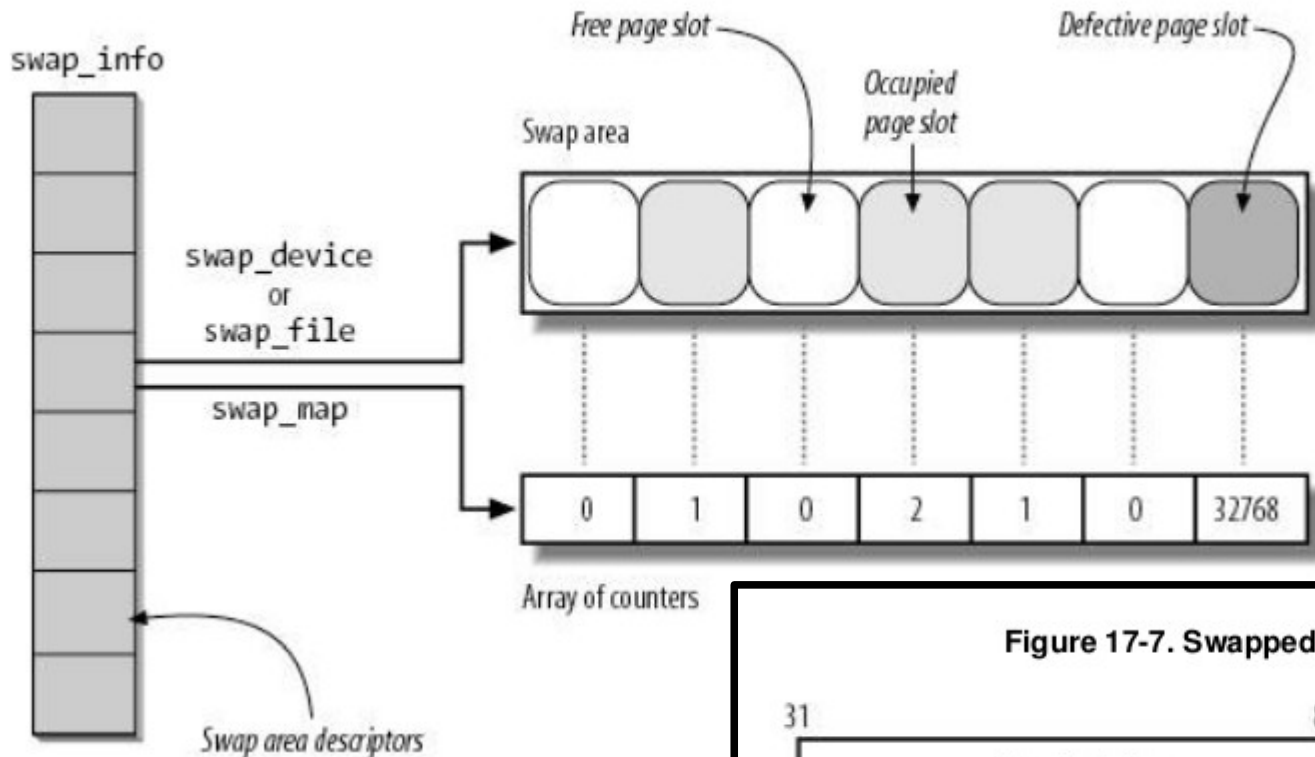
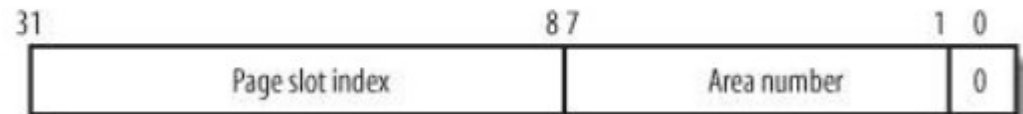


Figure 17-7. Swapped-out page identifier



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

