MEMORIA VIRTUAL

DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

UNIVERSIDAD NACIONAL DEL SUR

KMC © 2020

AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. Thrashing
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

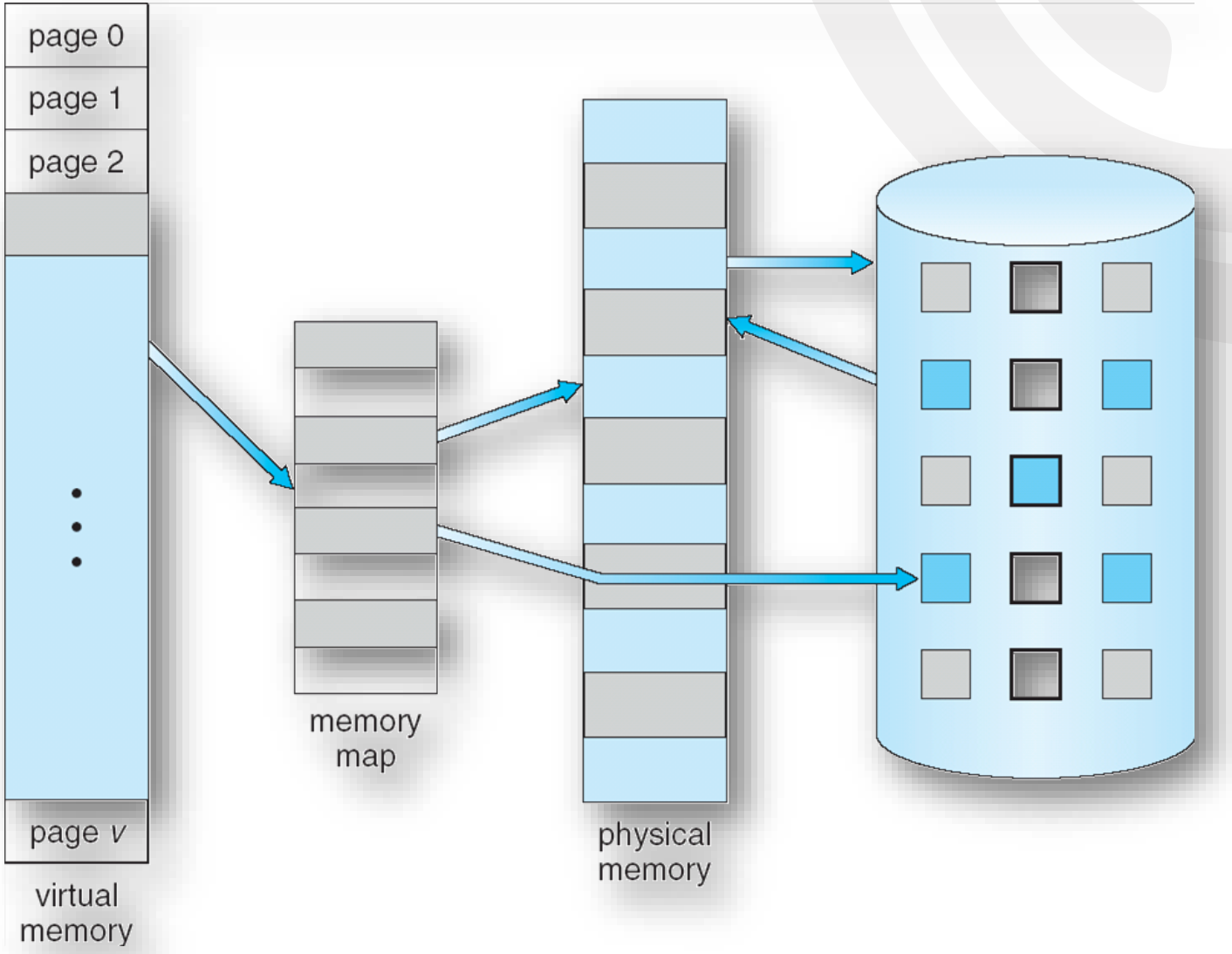
AGENDA

1. **Conceptos generales**
2. Demanda de Páginas
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. Thrashing
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

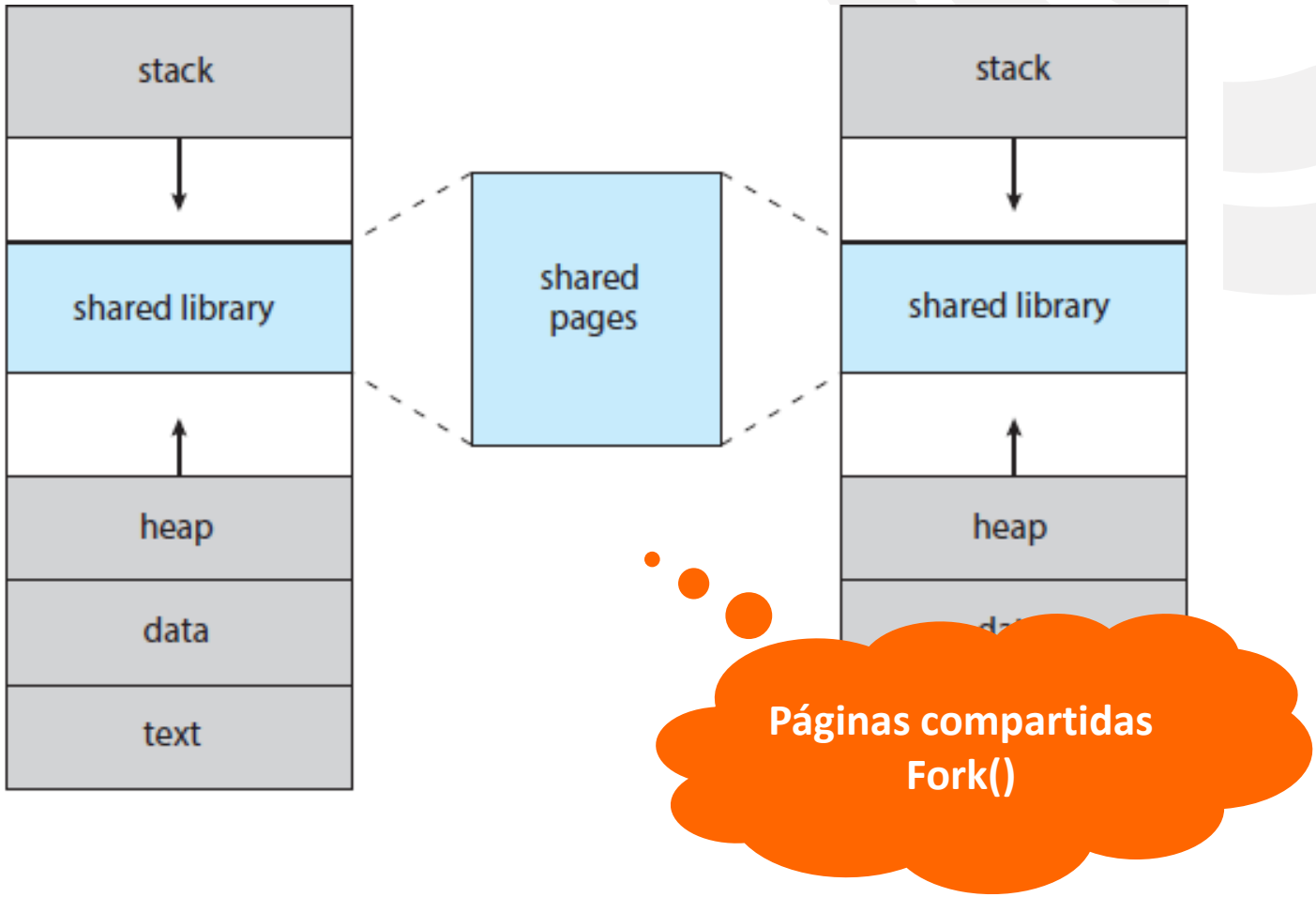
CONCEPTOS GENERALES

- **Memoria Virtual**– separación de la memoria lógica del usuario de la memoria física.
  - Solo parte del programa necesita estar en memoria para su ejecución.
  - El espacio de direcciones lógicas puede ser más grande que el espacio de direcciones físicas.
  - Permite a varios procesos compartir el espacio de direcciones.
  - La creación de procesos sea más eficiente
- La memoria virtual puede ser implementada vía:
  - Paginado por demanda
  - Segmentación por demanda

MEMORIA VIRTUAL MÁS GRANDE QUE LA MEMORIA FÍSICA



MEMORIA VIRTUAL Y LIBRERÍAS COMPARTIDAS



AGENDA

1. Conceptos generales
2. **Demanda de Páginas**
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. Thrashing
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

PAGINADO POR DEMANDA

- Traer una página a la memoria solo cuando es necesario.
  - Son necesarias menos E/S
  - Es necesaria menos memoria
  - Respuesta más rápida
  - Más usuarios
- Cuando una página se necesita $\Rightarrow$ se la referencia
  - referencia inválida $\Rightarrow$ aborto
  - no está en memoria $\Rightarrow$ se la trae a memoria
- Intercambiador “perezoso” – nunca intercambia en memoria hasta que la página se necesite.
  - El intercambiador (swapper) que trata con páginas es un *paginador* (pager)

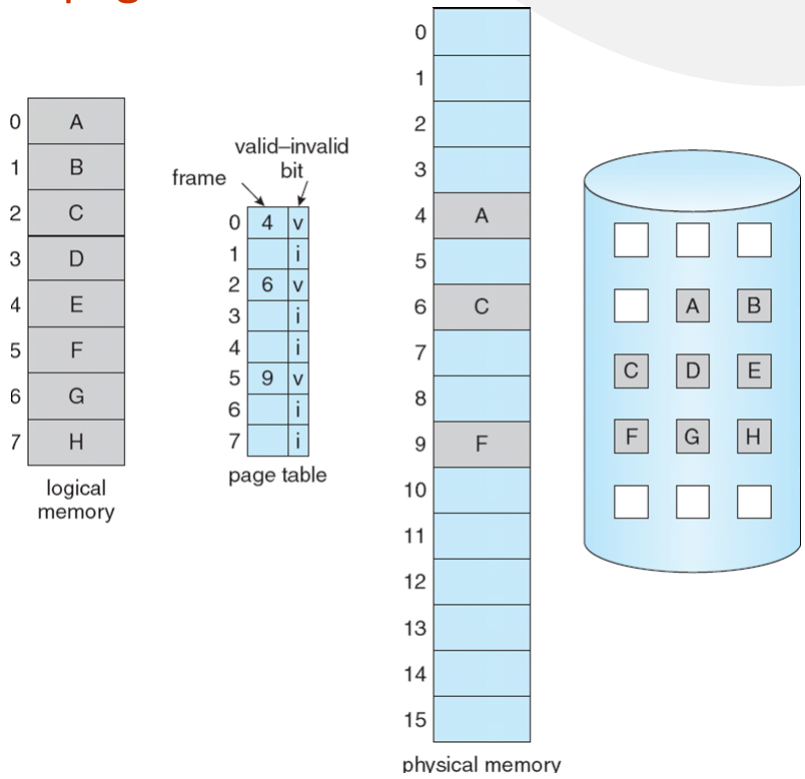
PAGINACIÓN: BIT VÁLIDO-INVÁLIDO

- Se asocia a cada entrada a la tabla de páginas un bit válido–inválido (1 ⇒ en memoria, 0 ⇒ no en memoria)
- Inicialmente el bit válido–inválido es puesto a 0 en todas las entradas.

  Durante la traducción de la dirección, si el bit válido–inválido en la entrada de la tabla de páginas es 0 ⇒ falta de página.

marco #	bit válido-inválido
	1
	1
	1
	1
	0
⋮	
	0
	0

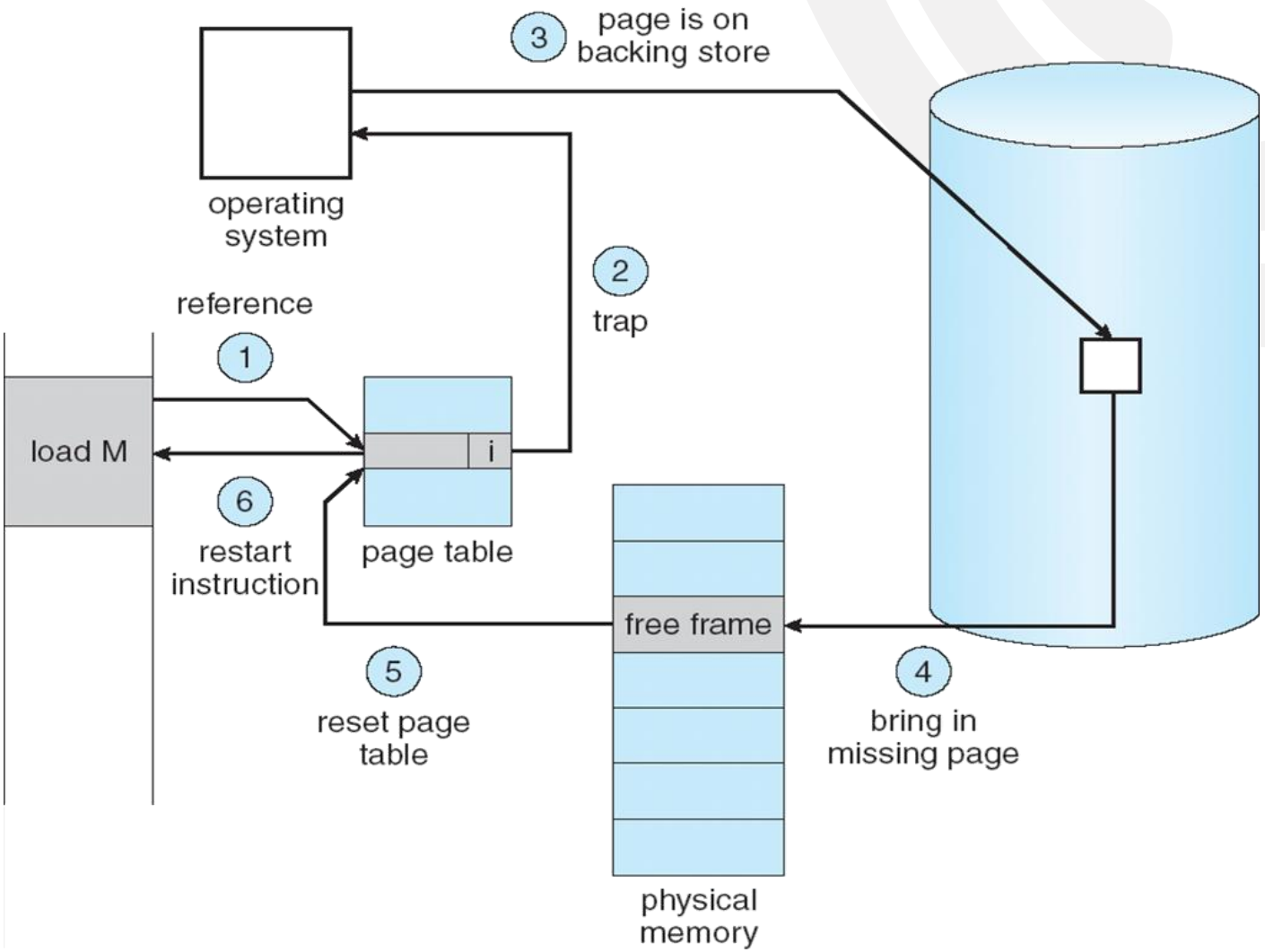
tabla de páginas



FALTA DE PÁGINA

- Si hay una referencia a una página, la primer referencia hace un **trap** al SO ⇒ **falta de página**
- El SO mira la tabla para decidir:
  - Referencia Inválida ⇒ aborto.
  - No está en memoria.
- Toma un marco libre.
- Lleva la página al marco.
- Reestablece las tablas, bit de validación = 1.
- Reinicio de la instrucción:
  - Movimiento de bloque
  - Locación con auto incremento/decremento

PASOS EN EL MANEJO DE UNA FALTA DE PÁGINA



PERFORMANCE DEL PAGINADO POR DEMANDA

- Ritmo de falta de páginas $0 \leq p \leq 1.0$
  - si $p = 0$ no hay falta de páginas
  - si $p = 1$, cada referencia es una falta de página
- Tiempo Efectivo de Acceso (TEA)

  TEA = (1 – *p*) x acceso a memoria + *p* x (sobrecarga de falta de página + salida de la página + entrada de la página + sobrecarga de reinicio)

Actividades

- Atender la interrupción
- Leer la página
- Reiniciar el proceso

PERFORMANCE DEL PAGINADO POR DEMANDA

Ejemplo

- Tiempo de acceso a memoria = 200 nanosegundos
- Tiempo promedio de servicio de una falta de página = 8 milisegundos.

$$TEA = (1 - p) \times 200 + p \times (8 \text{ milisegundos})$$

$$= (1 - p) \times 200 + p \times 8000000$$

$$= 200 + p \times 7999800$$

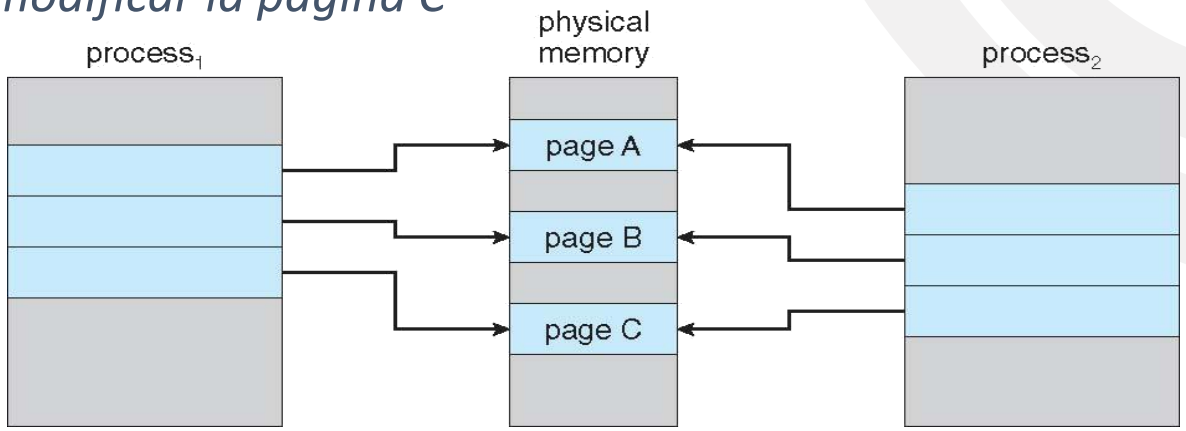
- Si uno de 1000 causa una falta de página, entonces TEA = 8.2 Microseconds.

Esto significa una reducción de 40!!

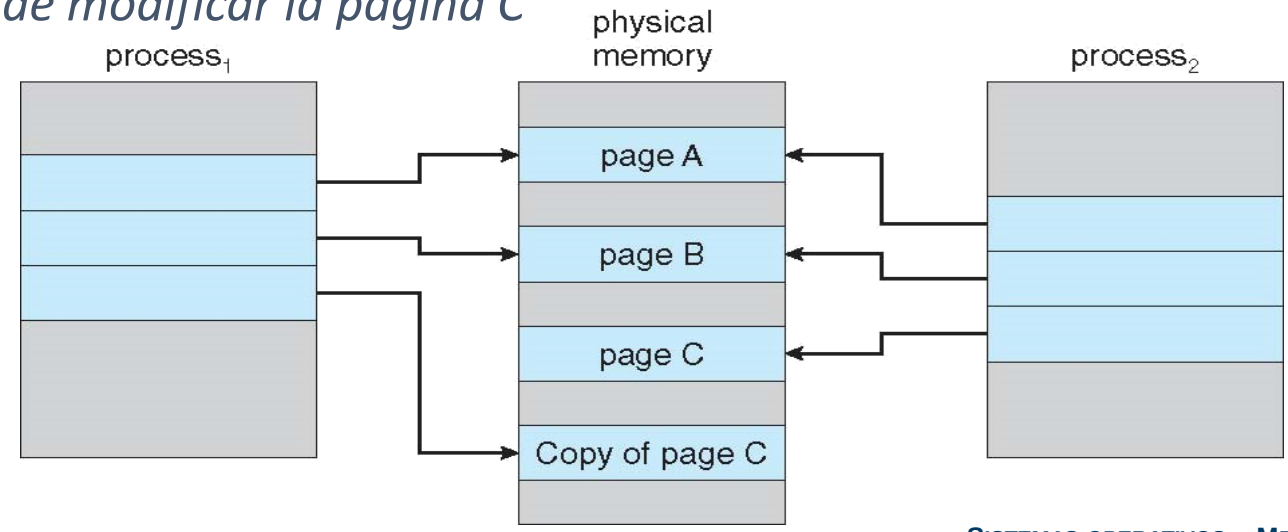
COPIA EN ESCRITURA

Permite que un proceso padre comparta con el proceso hijo (fork())

Antes de modificar la página C



Después de modificar la página C



AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. **Reemplazo de Páginas**
4. Alocación de Cuadros
5. Thrashing
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

¿QUÉ OCURRE CUANDO NO HAY MARCOS LIBRES?

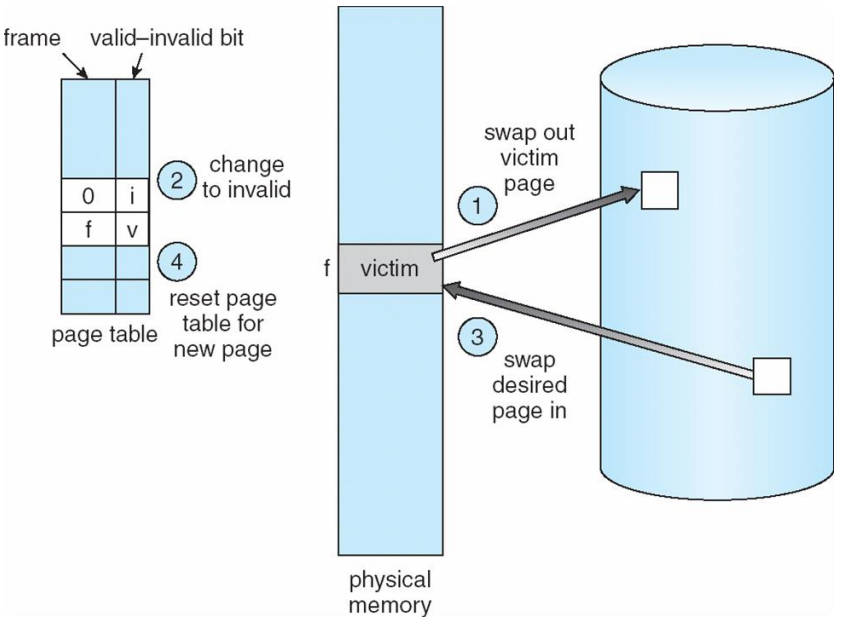
- Reemplazo de páginas – se busca alguna página en memoria que no está en uso y se la intercambia.
  - algoritmo
  - performance – se requiere un algoritmo que resulte en un mínimo número de falta de páginas.
- Algunas páginas pueden ser quitadas o volcadas en memoria varias veces.

REEMPLAZO DE PÁGINAS

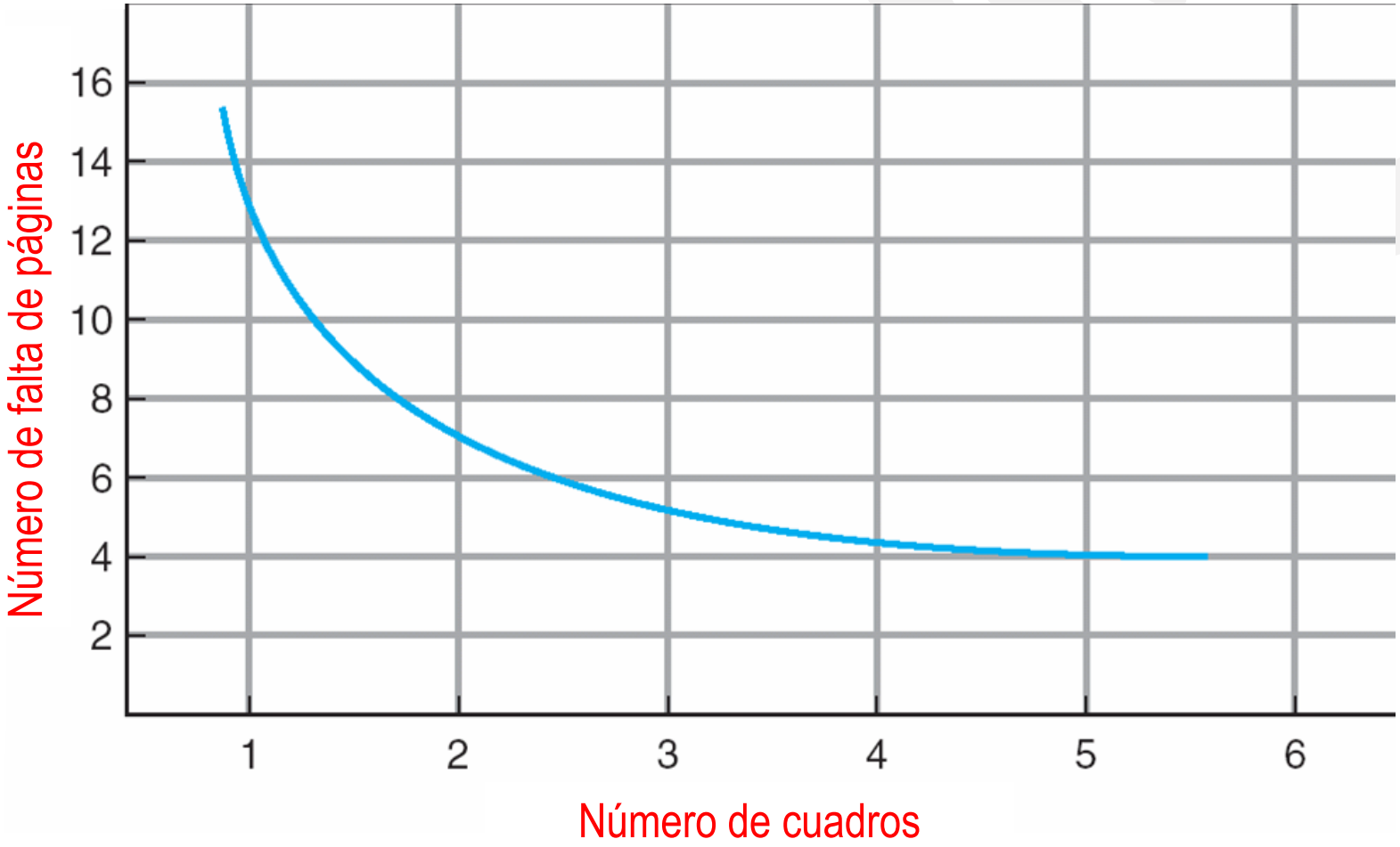
- Se previene sobrealocación de memoria por modificación de la rutina de servicio de falta de páginas para incluir el reemplazo de páginas.
- Uso del *bit de modificado* (*"sucio")* para reducir la sobrecarga de la transferencia de páginas – solo las páginas modificadas son escritas en disco.
- El reemplazo de páginas completa la separación entre memoria lógica y memoria física – puede ser provista una gran memoria lógica en una pequeña memoria física.

REEMPLAZO DE PÁGINAS

1. Encontrar la locación de la página deseada en el disco
2. Encontrar un marco libre:
   - Si hay un marco libre, usarlo
   - Si no hay marco libre usar un algoritmo de reemplazo de página para seleccionar el marco **víctima.**
3. Traer la página deseada al marco libre, modificar la tabla de páginas y la tabla de marcos.
4. Reiniciar el proceso



GRAFO DE FALTA DE PÁGINAS VERSUS NÚMERO DE CUADROS



ALGORITMOS DE REEMPLAZO DE PÁGINAS

- Procuran un ritmo de falta de páginas bajo.
- Se evalúa el algoritmo ensayándolo sobre una secuencia particular de referencias a memoria (secuencia de referencia) y computando el número falta de páginas en la secuencia.
- En el ejemplo, la secuencia es

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

ALGORITMO PRIMERO EN ENTRAR—PRIMERO EN SALIR (FIFO)

- Secuencia de referencia: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**
- 3 marcos (hay 3 páginas en memoria al mismo tiempo por proceso)

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

9 faltas de páginas

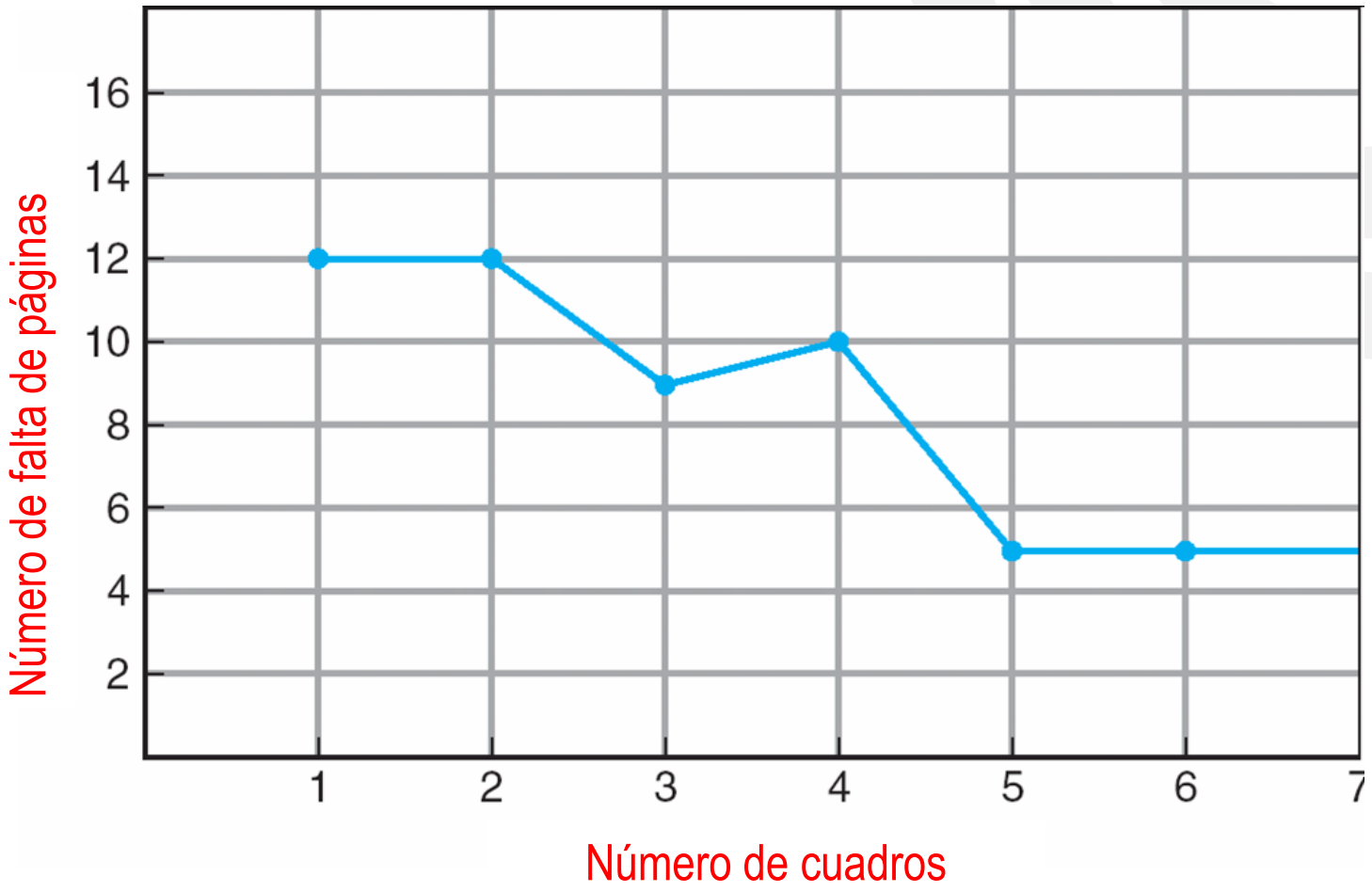
- 4 marcos

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

10 faltas de páginas

- Reemplazo FIFO – Anomalía de Belady
  - más marcos $\Rightarrow$ menos faltas de páginas

ANOMALÍA DE BELADY



ALGORITMO ÓPTIMO

- Reemplace la página que no será usada por un período largo de tiempo.
- Ejemplo con 4 marcos: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**

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

6 faltas de páginas

- ¿Cómo se conoce esto?
- Usado para medir como se comporta un algoritmo.

ALGORITMO *MENOS RECIENTEMENTE USADO* (LRU)

- Secuencia de referencia: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

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

8 faltas de páginas

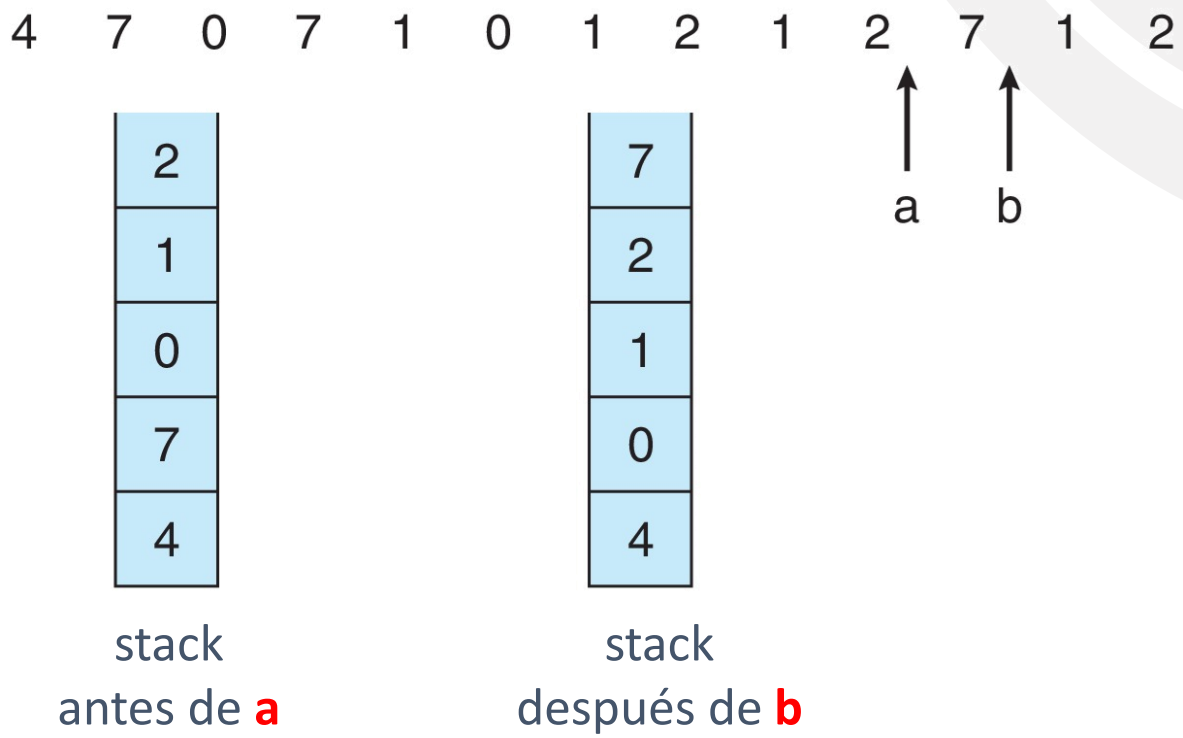
- Implementación del contador
  - Cada entrada a la tabla de páginas tiene un contador; cada vez que la página es referenciada se copia el reloj en el contador.
  - Cuando la página necesita ser cambiada, mira los contadores para determinar cuales hay que cambiar.

ALGORITMO LRU

- Implementation por Stack – mantiene un stack de números de páginas en forma de una lista doblemente enlazada:
  - Página referenciada:
    - se mueve al tope
    - Requiere cambios de punteros
  - No se necesita buscar para realizar el reemplazo

USO DEL STACK PARA REGISTRAR LAS REFERENCIAS A PÁGINAS MÁS RECIENTES

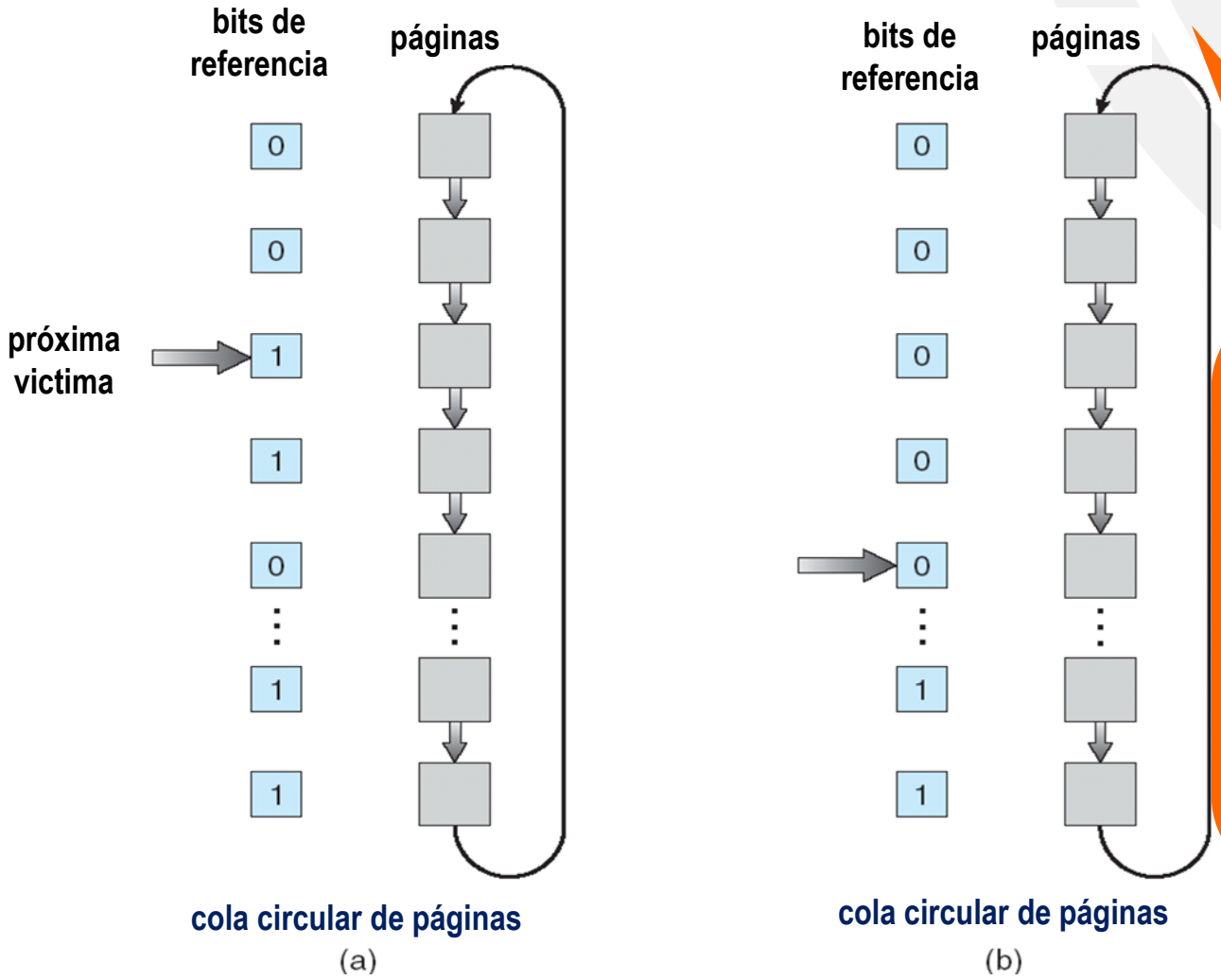
Secuencia de referencia



ALGORITMOS DE APROXIMACIÓN A LRU

- Bit de referencia. Con cada página se asocia un bit, inicialmente= 0
  - Cuando la página es referenciada el bit es puesto a 1.
  - Reemplace aquella en la cual es 0 (si existe). No se conoce el orden.
- Segunda oportunidad
  - Necesita el bit de referencia.
  - Reemplazo circular (Clock).
  - Si la página a ser reemplazada (en orden circular) tiene el bit de referencia = 1 entonces:
    - Se pone el bit de referencia en 0.
    - Se deja la página en memoria.
    - Se reemplaza la siguiente página (en el orden circular), sujeta a las mismas reglas.

ALGORITMO DE REEMPLAZO DE PÁGINAS SEGUNDA OPORTUNIDAD (RELOJ)



Una mejora de este algoritmo es considerar el bit de referencia y el bit de modificado.

1.- (0, 0)
2.- (0, 1)
3.- (1, 0)
4.- (1, 1)

ALGORITMOS DE CUENTA

- Se mantiene un contador del número de referencias que han sido hechas a la misma página.
- **Algoritmo LFU**: reemplaza la página con la menor cuenta.
- **Algoritmo MFU**: reemplaza la página con la mayor cuenta, basado en el argumento que la página con la cuenta más chica fue recién puesta y todavía tiene que ser usada.

AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. Reemplazo de Páginas
4. **Alocación de Cuadros**
5. Thrashing
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

ALOCACIÓN DE MARCOS

- Cada proceso necesita un número mínimo de páginas, definido por la arquitectura de la computadora.
- Por ejemplo: IBM 370 – 6 páginas son necesarias para manejar la instrucción SS MOVE:
  - la instrucción es de 6 bytes, puede expandirse a 2 páginas.
  - 2 páginas para manejar **desde**.
  - 2 páginas para manejar **hacia**.
- Dos esquemas de alocación.
  - alocación fija
  - alocación con prioridad

ALOCACIÓN FIJA

- Alocación igualitaria – p. e., si hay 100 marcos y 5 procesos, a cada uno se les da 20 páginas.
- Alocación proporcional – aloca de acuerdo al tamaño del proceso.

S_i = tamaño del proceso P_i
$S = \Sigma S_i$
m = número de marcos
a_i = alocación para $p_i = S_i/S \times m$

m = 64
S1 = 10
S2 = 127
a1 = 10 / 134 x 64 ≈ 5
a2 = 127 / 134 x 64 ≈ 59

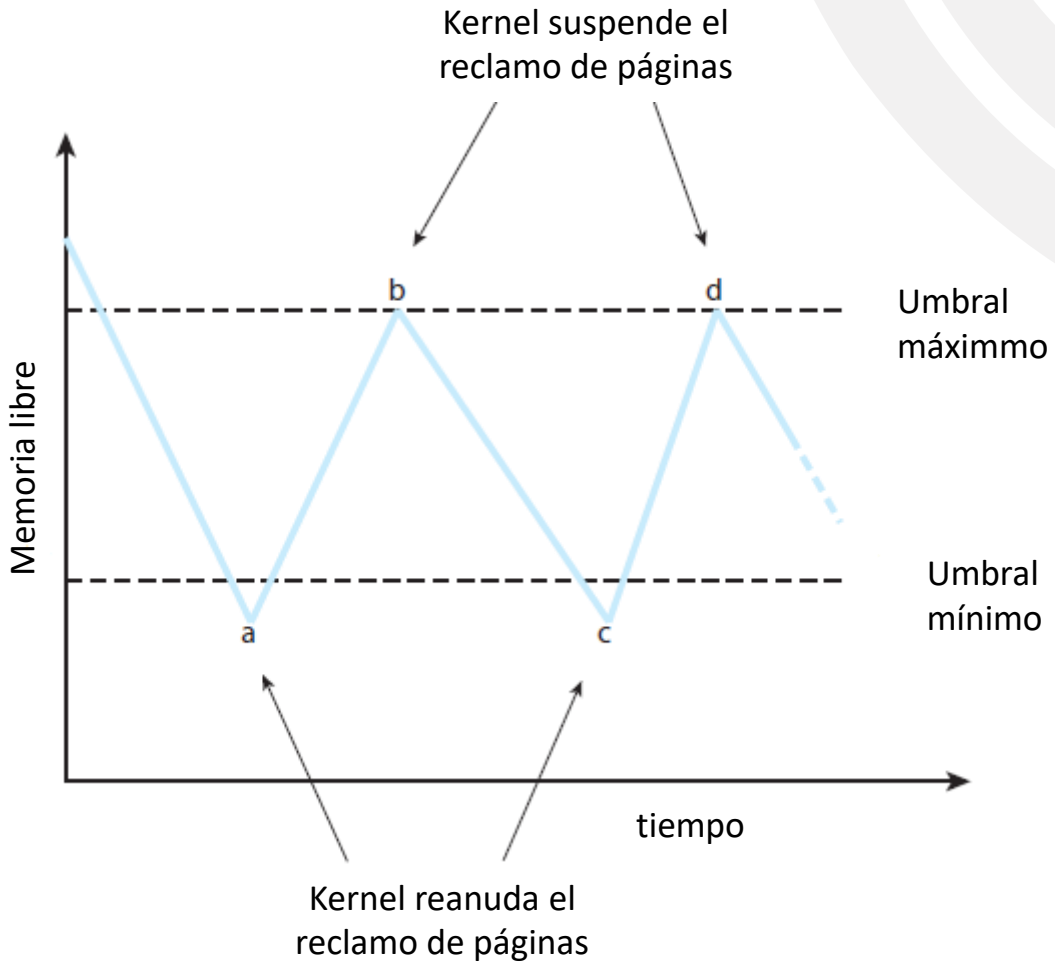
ALOCACIÓN CON PRIORIDAD

- Se usa un esquema de alocación proporcional usando prioridades antes que tamaño.
- Si el proceso P_i genera una falta de página
  - Se selecciona para reemplazar uno de sus marcos.
  - Se selecciona para reemplazar un marco de un proceso con menor número de prioridad.

REEMPLAZO GLOBAL VS. LOCAL

- Reemplazo global – el proceso selecciona un marco de reemplazo de todos los marcos; un proceso puede tomar los marcos de otro.
- Reemplazo local – cada proceso selecciona de su propio conjunto el marco a reemplazar.

REEMPLAZO GLOBAL VS. LOCAL

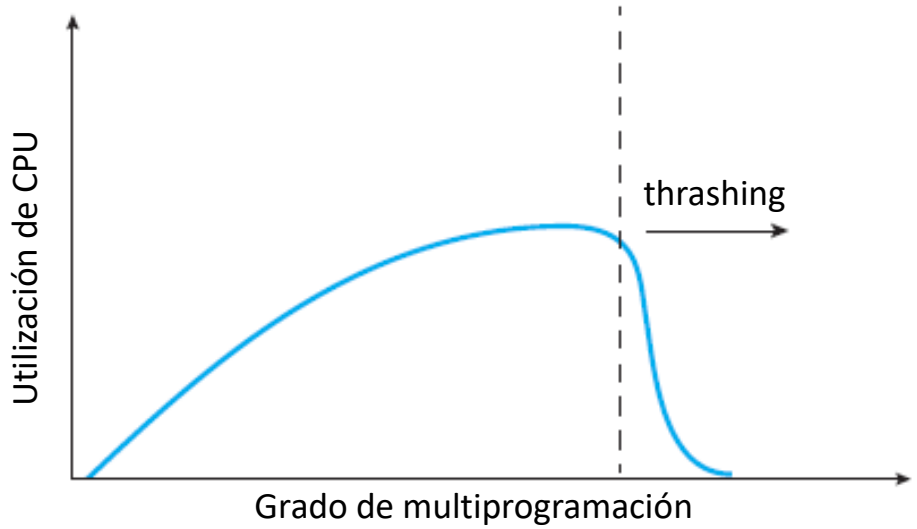


AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. **Thrashing**
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

THRASHING

- Si un proceso no tiene suficientes páginas, el ritmo de falta de páginas es muy alto. Esto lleva a:
  - baja utilización de CPU.
  - el SO piensa que es necesario incrementar el grado de multiprogramación.
  - otro proceso se agrega al sistema.
- Thrashing ≡ un proceso está ocupado haciendo solamente intercambio de páginas.



THRASHING

- ¿Por qué trabaja el paginado?
  Modelo de Localidad
  - El proceso migra desde una localidad a otra.
  - Las localidades se pueden solapar.
- ¿Por qué ocurre el thrashing ?
  Σ tamaño de la localidad > tamaño total de memoria

AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. Thrashing
6. **Conjunto de trabajo (working-set)**
7. Otras Consideraciones
8. Ejemplos

MODELO DE CONJUNTO DE TRABAJO (WORKING-SET)

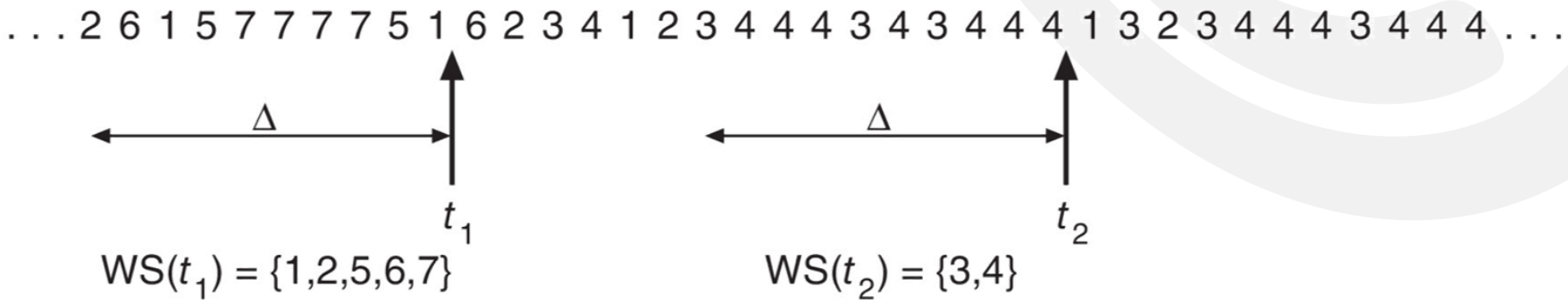
El modelo está basado en la localidad.

- $\Delta \equiv$ ventana working-set $\equiv$ un número fijo de referencias de páginas. Ejemplo: 10,000 instrucciones
- WSS_i (working set del proceso P_i) = número total de páginas referenciadas en el más reciente Δ (varía en el tiempo)
  - $\Rightarrow$ si Δ es demasiado chico no acompaña la localidad.
  - $\Rightarrow$ si Δ es demasiado grande acompaña varias localidades.
  - $\Rightarrow$ si $\Delta = \infty \Rightarrow$ acompañará al programa entero.
- $D = \Sigma\ WSS_i \equiv$ demanda total de marcos
- si $D > m \Rightarrow$ Thrashing
- Política: si $D >$ m, entonces suspende uno de los procesos.

m=# de marcos de memoria

MODELO DE WORKING-SET

Referencias a la tabla de páginas

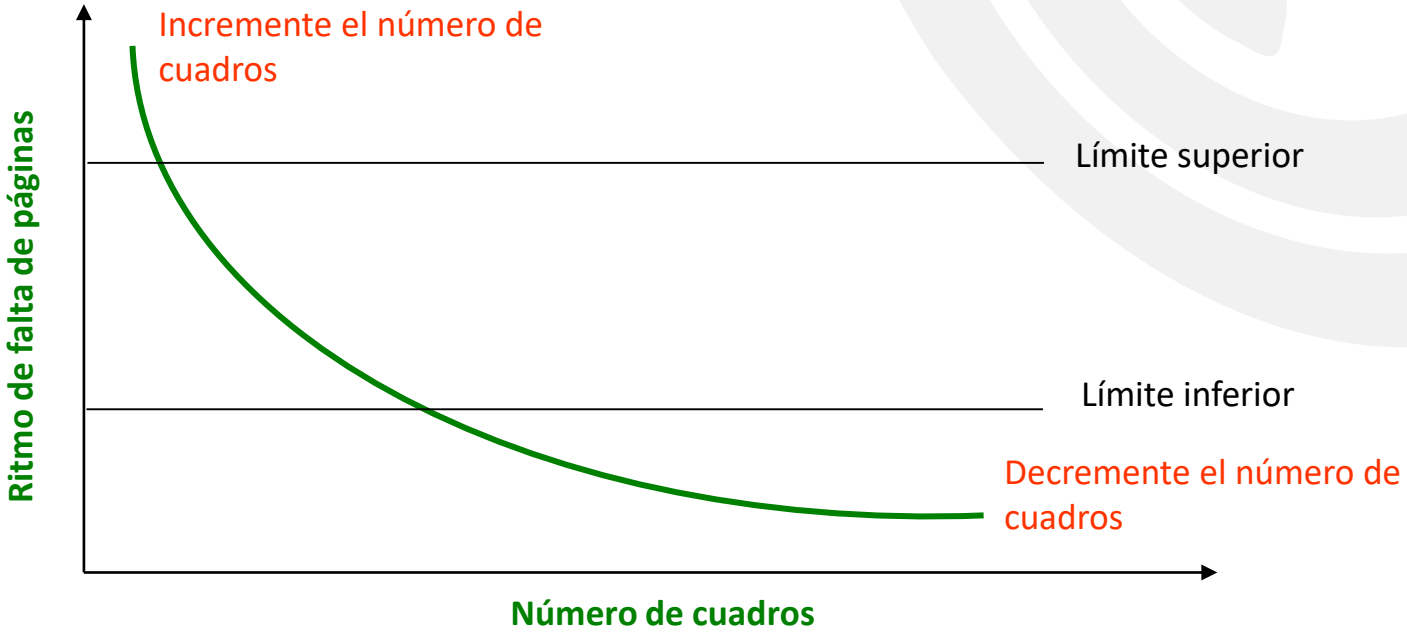


Δ = 10

CONTROL DEL WORKING SET

- Aproximar con un intervalo de tiempo + bit de referencia
- Ejemplo: Δ = 10,000
  - Las interrupciones del Timer se producen cada 5000 unidades de tiempo.
  - Se mantienen en memoria 2 bits por cada página.
  - Siempre que el timer interrumpe copia e inicializa los valores de todos los bits de referencia a 0.
  - Si uno de los bits en memoria = 1 $\Rightarrow$ página en el working set.
- ¿Por qué no es completamente preciso?
- Mejora = 10 bits e interrupción cada 1000 unidades de tiempo.

ESQUEMA DE FRECUENCIA DE FALTA DE PÁGINA



- La idea es establecer un ritmo “aceptable” de falta de páginas.
  - Si el ritmo actual es demasiado bajo, los procesos pierden marcos.
  - Si el ritmo actual es demasiado alto, el proceso gana marcos.

AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. Thrashing
6. Conjunto de trabajo (working-set)
7. **Otras Consideraciones**
8. Ejemplos

OTRAS CONSIDERACIONES - PREPAGINADO

PREPAGINADO

- Para reducir el gran número de falta de páginas que ocurren en el inicio del proceso
- Se necesitará prepaginar todas o algunas páginas del proceso antes de ser referenciadas
- Pero si las páginas prepaginadas no son usadas se incurrió en gasto de E/S y memoria
- Suponga que ***s*** páginas son prepaginadas y ***α*** de esas páginas son usadas
  - Es el costo de salvar ***s* α*** faltas > o < qué el costo de prepaginar ***s* (1- α)*** páginas no necesarias?
  - ***α*** cercano a cero ⇒ prepaginado pierde

OTRAS CONSIDERACIONES – TAMAÑO DE PÁGINA

SELECCIÓN DEL TAMAÑO DE PÁGINA

- fragmentación
- tamaño de tabla
- sobrecarga de E/S
- localidad

OTRAS CONSIDERACIONES – ESTRUCTURA DEL PROGRAMA

- Estructura de programa

  Arreglo de enteros A[1024, 1024]

  Cada fila está almacenada en una página

  Un cuadro

  - Programa 1

```
for j := 1 to 1024 do
    for i := 1 to 1024 do
        A[i,j] := 0;
```

  1024 x 1024 faltas de páginas

  - Programa 2

```
for i := 1 to 1024 do
    for j := 1 to 1024 do
        A[i,j] := 0;
```

  1024 faltas de páginas

- Fijación para E/S y direccionamiento

OTRAS CONSIDERACIONES – FIJACIÓN DE E/S

- **Fijación de E/S** – Algunas veces las páginas deben ser fijadas en la memoria

- Considere E/S – Las páginas que son usadas para copiar un archivo desde un dispositivo deben ser fijadas para no ser seleccionadas por el algoritmo de reemplazo de páginas

AGENDA

1. Conceptos generales
2. Demanda de Páginas
3. Reemplazo de Páginas
4. Alocación de Cuadros
5. Thrashing
6. Conjunto de trabajo (working-set)
7. Otras Consideraciones
8. Ejemplos

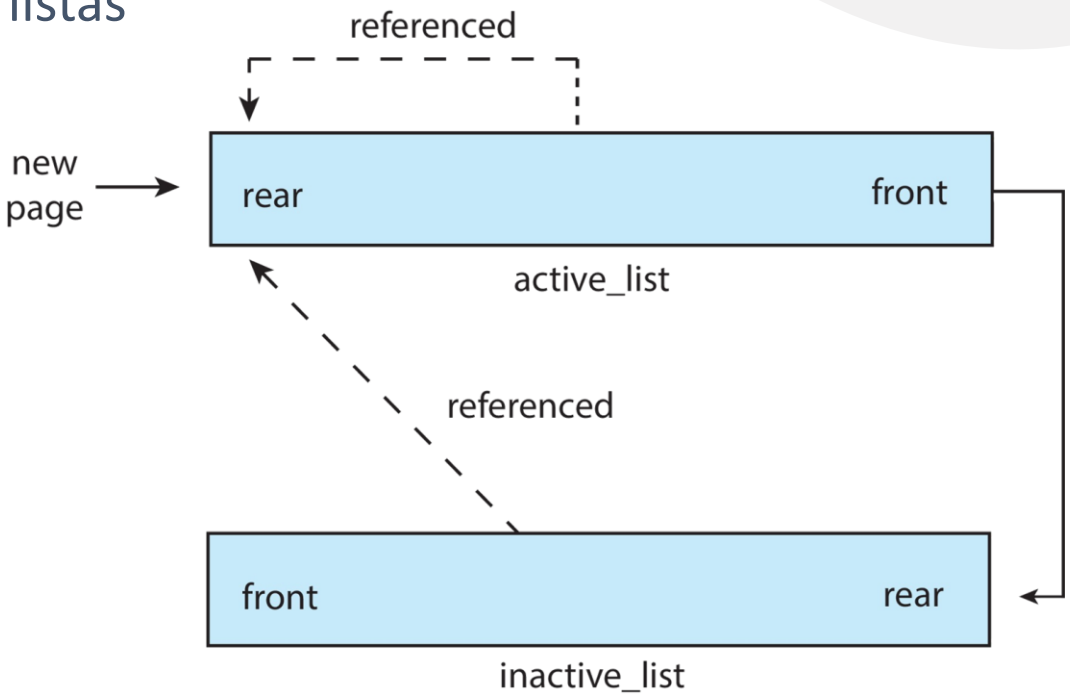
EJEMPLOS

Administración de memoria virtual

- Linux
- Windows
- UNIX-SVR4
- Solaris

LINUX

- Usa demanda de páginas, aloca las páginas de una lista de marcos libres.
- Usa política de reemplazo global, similar a la aproximación de LRU utilizando el algoritmo del Reloj.
- Mantiene dos tipos de listas
  - Lista Activa
  - Lista Inactiva



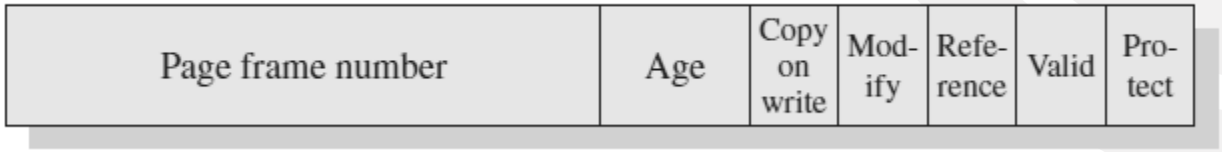
WINDOWS

- Demanda de páginas con **clustering**. Es decir trae las páginas que rodean a la página demandada.
- Política de reemplazo global o local, similar a la aproximación de LRU utilizando el algoritmo del Reloj.
- Mantiene una lista de marcos libres.
- A los procesos se les asigna un **working set mínimo** y un **working set máximo.**
- Cuando la cantidad de memoria libre en el sistema cae bajo determinado umbral, se activa en forma automática hasta restaurar la cantidad de memoria libre.
- Este ajuste automático remueve páginas de los procesos que están excedidos de su *working set* mínimo.

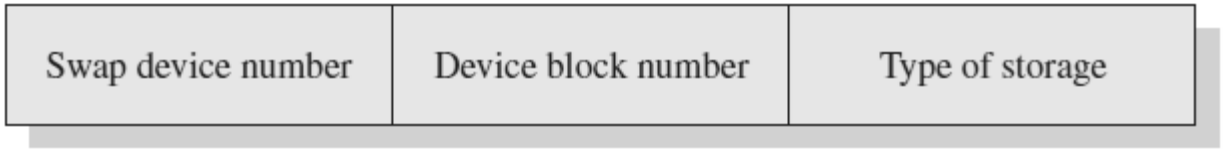
UNIX-SVR4

Formato Administración de Memoria

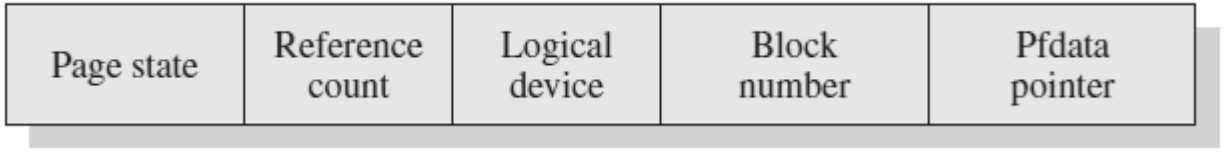
Entrada en la tabla de páginas



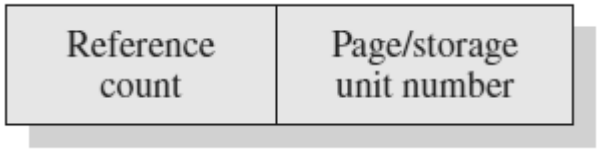
Descriptor de bloque de disco



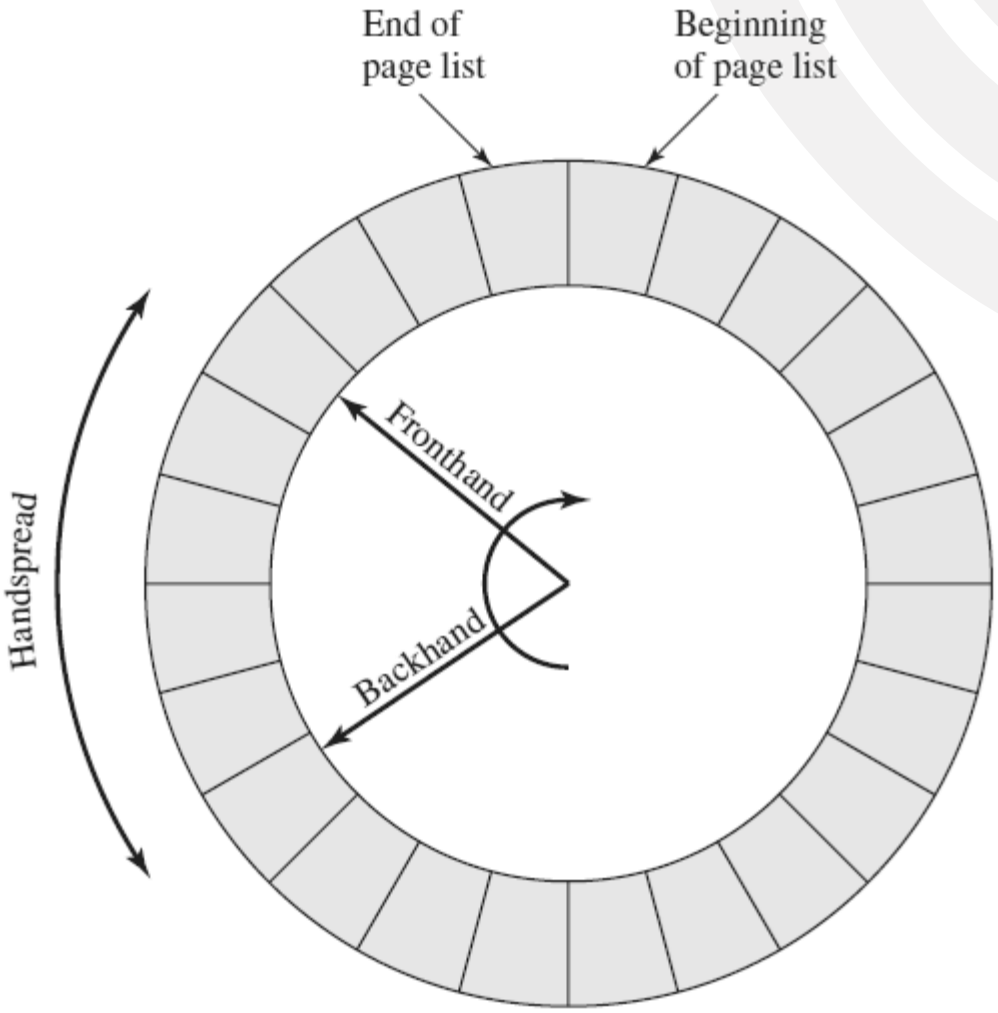
Entrada en la tabla de dato de página frame



Entrada en la tabla de swap-use



UNIX- SVR4: ALGORITMO REEMPLAZO DE PÁGINA BASADO EN EL RELOJ



UNIX-SOLARIS-LINUX

- Mantiene una lista de páginas libres para asignar a procesos en falta
- ***Lotsfree*** – parámetro umbral (cantidad de memoria libre) para comenzar a paginar
- ***Desfree*** – parámetro umbral para incrementar el paginado
- ***Minfree*** – parámetro umbral para ser intercambiadas las páginas
- El paginado es realizado por un proceso ***pageout***
- Pageout barre las páginas usando un algoritmo de reloj modificado
- ***Scanrate*** es la frecuencia con que las páginas son barridas. Estos rangos varían entre ***slowscan*** y ***fastscan***
- La frecuencia de llamado a *pageout* depende de la cantidad de memoria libre disponible.

Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "Operating System Concepts", 7ma Edición 2009, 9na Edición 2012, 10ma Edición 2018.
- Stallings, W. "Operating Systems: Internals and Design Principles", Prentice Hall, 6ta Edición 2009, 7ma Edición 2011, 9na Edición 2018.