Principios de Sistemas Operativos

Memoria Virtual

Armando Arce, Escuela de Computación, arce@itcr.ac.cr

Tecnológico de Costa Rica

Memoria Virtual

La técnica de memoria virtual es un mecanismo que permite la ejecución de procesos que no se encuentren completamente en memoria.

- Una de las principales ventajas de este esquema es que los programas pueden tener un tamaño mayor que la propia memoria física. Además, se pueden ejecutar más programas al mismo tiempo.

Fundamentos

La memoria virtual incluye la separación de la memoria lógica, tal como la percibe el usuario, con respecto a la memoria física. Esta separación permite proporcionar a los programadores una memoria virtual extremadamente grande, cuando sólo está disponible una memoria física de menor tamaño.

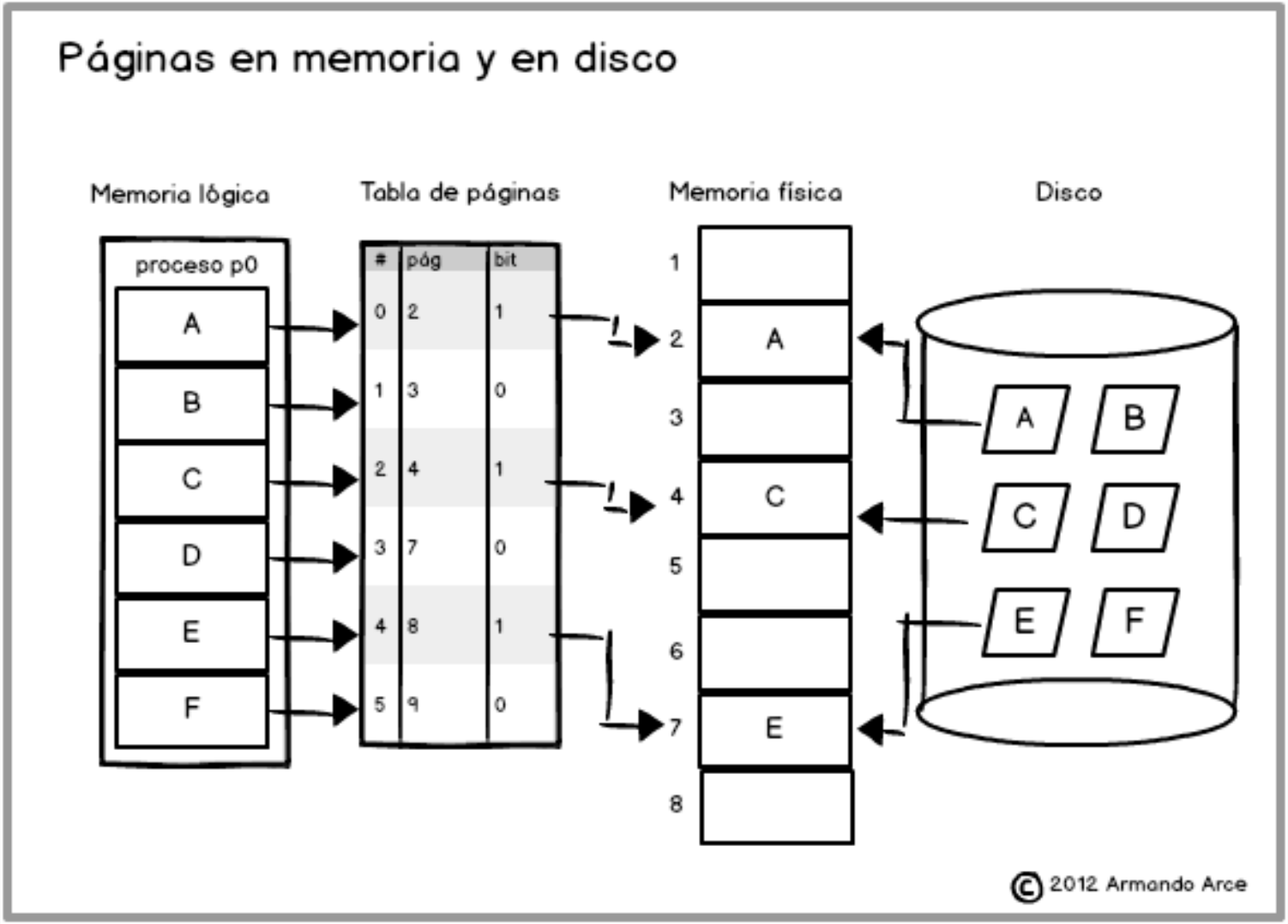
Páginas en memoria y en disco
Memoria lógica
Tabla de páginas
Memoria física
Disco
proceso p0
A
B
C
D
E
F
pág bit
0 2 1
1 3 0
2 4 1
3 7 0
4 8 1
5 9 0
1
2 A
3
4 C
5
6
7 E
8
A B
C D
E F
© 2012 Armando Arce

Figure 1:

Fundamentos

Además, la memoria virtual también permite que dos o más procesos compartan los archivos y la memoria mediante mecanismos de compartición de páginas. Esto proporciona las siguientes ventajas:

- Las bibliotecas del sistema pueden ser compartidas por numerosos procesos.
- Los procesos pueden compartir memoria.
- Se pueden compartir páginas durante la creación de procesos.

Paginación bajo demanda

La técnica de paginación bajo demanda carga inicialmente las páginas únicamente cuando sean necesarias, y se utiliza comúnmente en los sistemas de memoria virtual.

Paginación bajo demanda

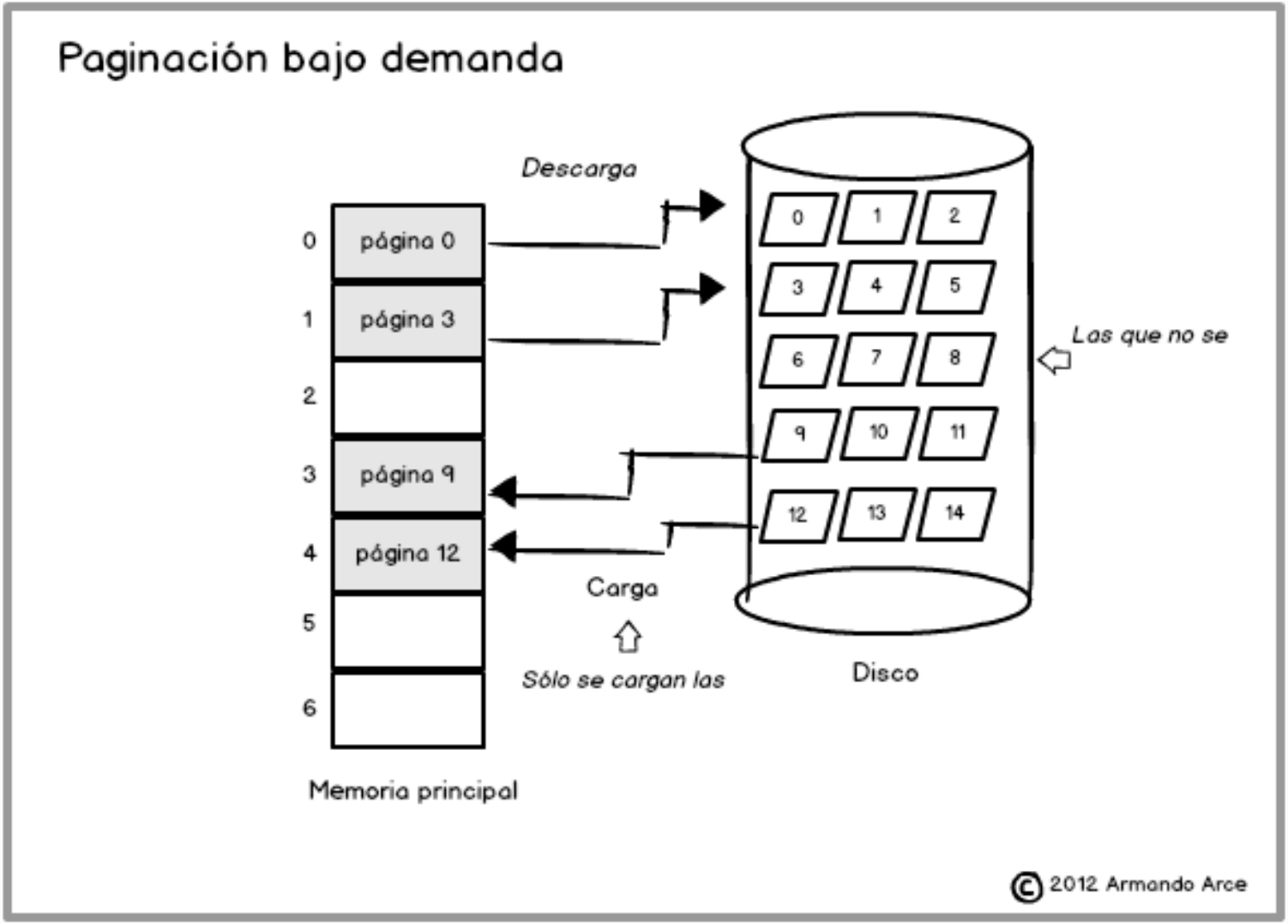

Figure 2:

Paginación bajo demanda

Un sistema de paginación bajo demanda es similar a un sistema de paginación con intercambio en el que los procesos residen en memoria secundaria (usualmente en disco).

- En este caso se utiliza un *intercambiador perezoso* que no intercambie una página con la memoria a menos que esa página vaya a ser necesaria.
- Sin embargo el término más correcto para referirse al *intercambiador* es *paginador*, pues el primero realiza el intercambio del proceso completo.

Conceptos básicos

Se requiere un soporte hardware para distinguir entre las páginas que se encuentran en memoria y las páginas que residen en el disco.

- Para esto se utiliza el bit válido/inválido.
- El acceso a una página marcada como inválida provoca una interrupción conocida como *fallo de página*.
- Esta interrupción se produce como resultado de que el sistema operativo no ha cargado anteriormente en memoria la página deseada.

Conceptos básicos

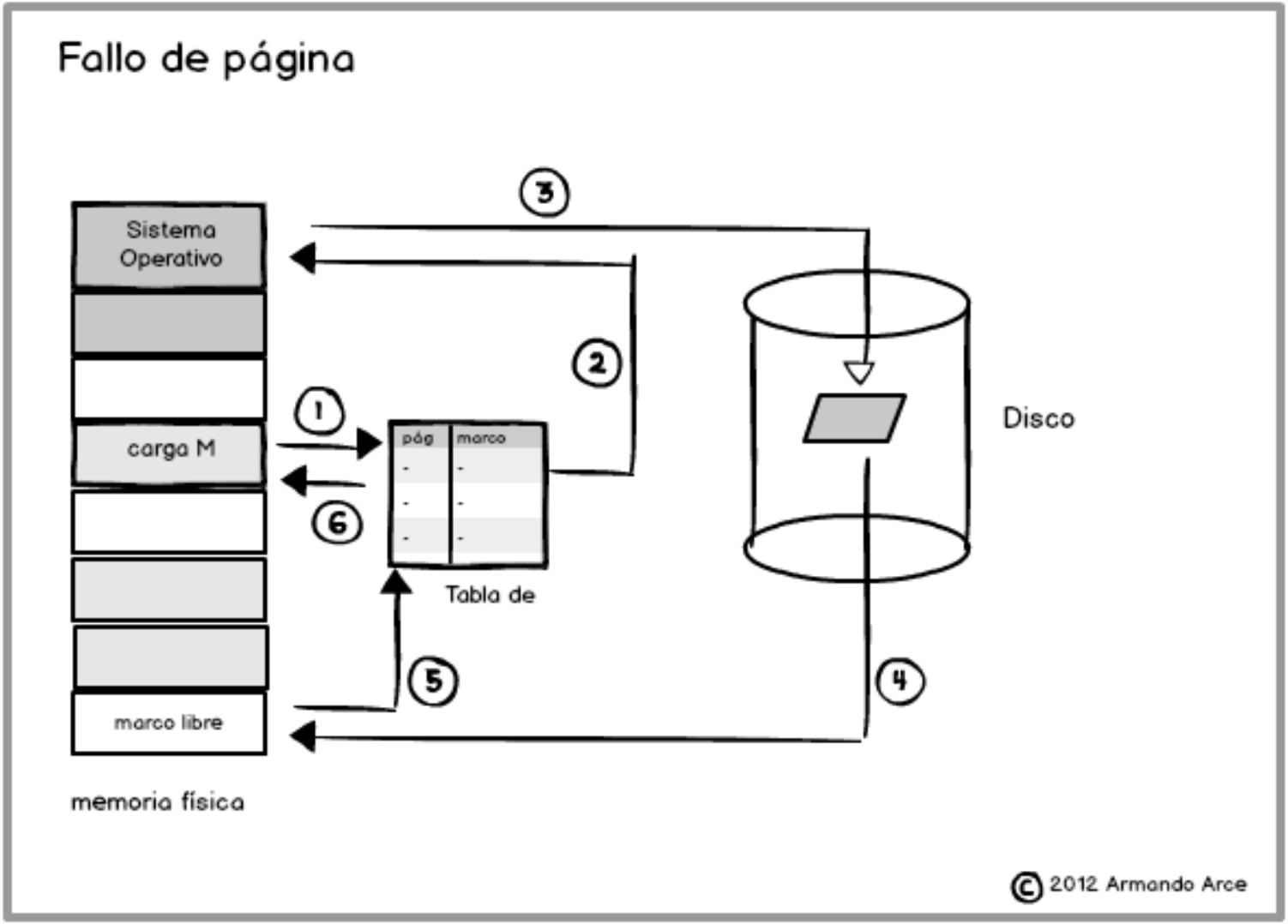

Figure 3:

Conceptos básicos

El procedimiento para tratar el fallo de página es el siguiente:

1. Se comprueba una tabla interna correspondiente al proceso, para determinar si la referencia era un acceso de memoria válido o inválido.
2. Si la referencia es inválida, se termina el proceso. Si era válida pero esa página todavía no ha sido cargada, se carga en memoria.
3. Se busca un marco libre:

3.1. Si hay un marco libre utilizarlo

3.2. Si no hay ningún marco libre, utilizar un algoritmo de sustitución de páginas para seleccionar un marco víctima

3.3. Escribir el marco víctima en el disco; cambiar las tablas de página y de marcos correspondientes

Conceptos básicos

4. Se ordena una operación de disco para leer la página deseada en el marco recién asignado.
5. Una vez completada la lectura de disco, se modifica la tabla interna que se mantiene con los datos del proceso y la tabla de páginas para indicar que dicha página se encuentra ahora en memoria.
6. Se reinicia la instrucción que fue interrumpida. El proceso podrá ahora acceder a la página como si siempre hubiera estado en memoria.

Copia durante la escritura

Se puede utilizar una técnica conocida con el nombre de *copia durante la escritura*, que funciona permitiendo que los procesos padre e hijo compartan inicialmente las mismas páginas.

- Estas páginas compartidas se marcan como páginas de copia durante la escritura, lo que significa que si cualquiera de los procesos escribe en una de las páginas compartidas, se creará una copia de esa página compartida.

Copia durante la escritura

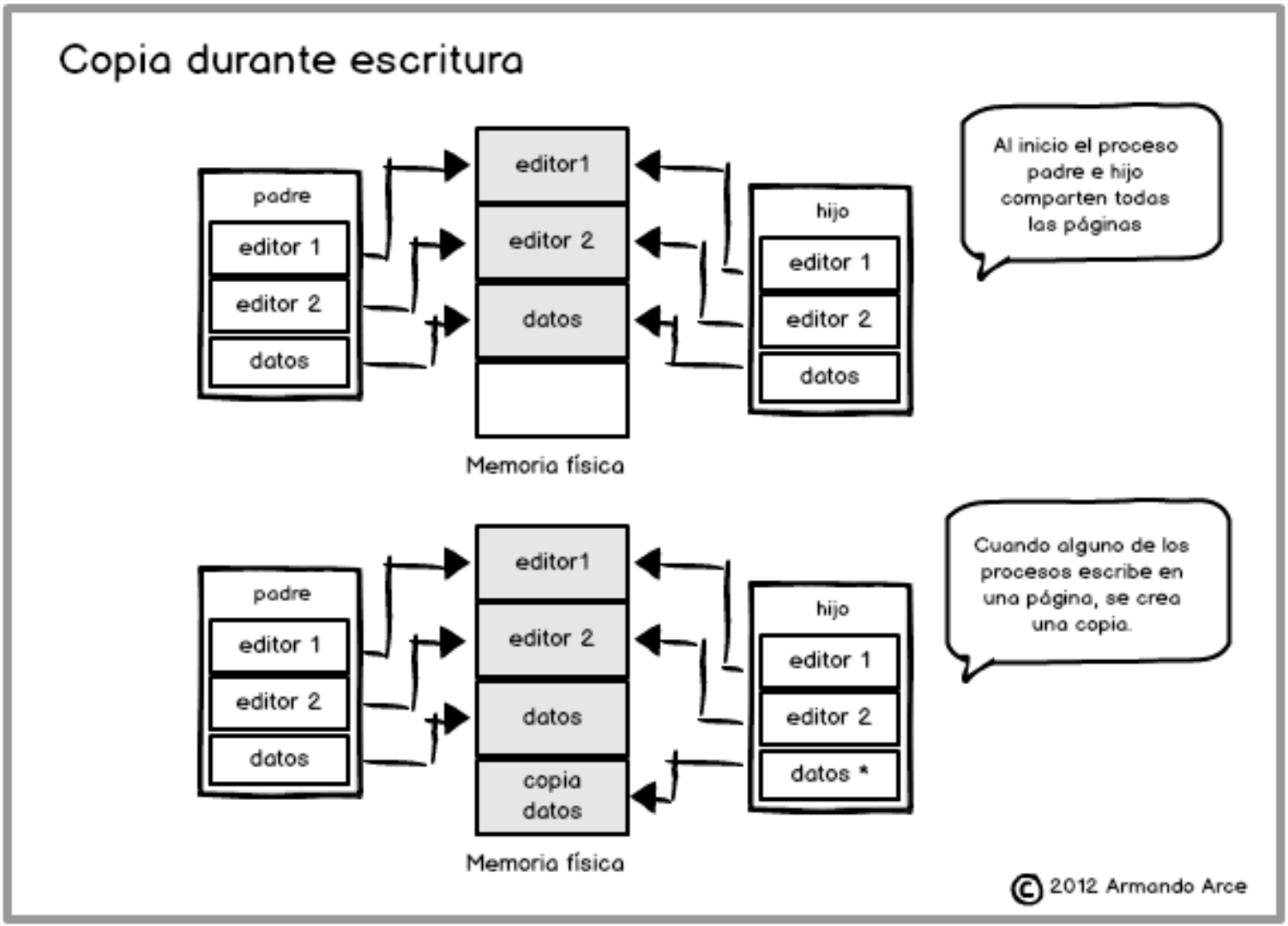

Figure 4:

Copia durante la escritura

Cuando se utiliza la técnica de copia durante la escritura, sólo se copian las páginas que sean modificadas por algunos de los procesos; todas las páginas no modificadas podrán ser compartidas por los procesos padre e hijo.

- Observe también que sólo es necesario marcar como de copia durante la escritura aquellas páginas que puedan ser modificadas.
- Las páginas que no puedan modificarse (páginas que contengan el código ejecutable) pueden compartirse entre el padre y el hijo.

Sustitución de páginas

Se debe tomar en cuenta que la memoria del sistema no se utiliza sólo para almacenar páginas de programas.

- Los búferes de E/S también consumen una cantidad significativa de memoria.
- Decidir cuánta memoria asignar a la E/S y cuánta a las páginas de programa representa un considerable desafío.
- Algunos sistemas asignan un porcentaje fijo de memoria para E/S, mientras otros permiten que los proceso de usuario y del subsistema de E/S compitan por la memoria del sistema.

Sustitución de páginas

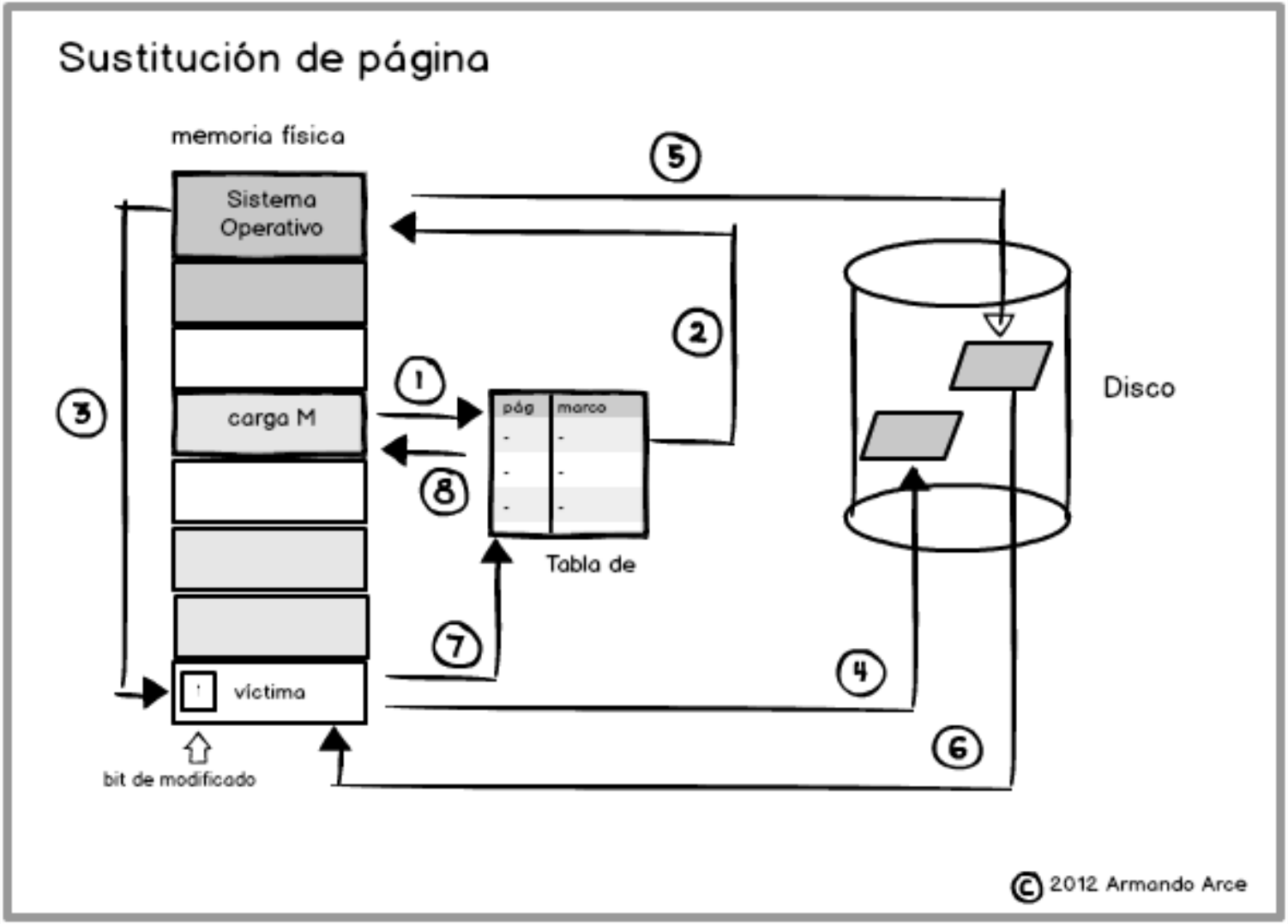

Figure 5:

Sustitución de páginas

La *sobreasignación de memoria* se da cuando se está ejecutando un proceso de usuario que produce un fallo de página y no existe ningún marco libre en memoria principal.

- El sistema operativo dispone de varias posibilidades en este punto. Una es terminar un proceso de usuario; sin embargo, esta opción no es la mejor.
- Otra posibilidad es descargar un proceso de memoria, liberando todos los marcos correspondientes y reduciendo el nivel de multiprogramación.
- Esta opción resulta adecuada en algunas circunstancias; no obstante, la solución más comúnmente utilizada es la *sustitución de páginas*.

Sustitución básica de páginas

La sustitución de páginas usa la siguiente técnica.

- Si no hay ningún marco libre, se localiza uno que no esté siendo actualmente utilizado y lo liberamos.
- Se puede liberar un marco escribiendo su contenido en el espacio de intercambio y modificando la tabla de páginas (y todas las demás tablas) para indicar que esa página ya no se encuentra en memoria.
- Ahora se puede utilizar el marco liberado para almacenar la página que provocó el fallo de página en el proceso.

Sustitución básica de páginas

Obsérvese que, si no hay ningún marco libre, se necesitan dos transferencias de páginas (una descarga y una carga).

- Esta situación duplica, en la práctica, el tiempo de servicio del fallo de página e incrementa correspondientemente el tiempo efectivo de acceso.
- Se puede reducir esta carga de trabajo adicional utilizando un *bit de modificación* (o bit sucio) que cada página o marco tiene asociado en el hardware.
- Este bit es activado cada vez que se escribe en la página, para indicar que se ha modificado.

Sustitución básica de páginas

Cuando se selecciona una página para sustitución, se examina el bit de modificación.

- Si el bit está activado, la página se debe escribir al disco pues fue modificada.
- Si el bit no está activado, no se necesita escribir la página en disco.
- Esto último pasa igual con las páginas de solo lectura, se pueden descartar cuando se desee.
- Este esquema reduce a la mitad el tiempo de E/S si la página no ha sido modificada.

Sustitución básica de páginas

El mecanismo de sustitución de páginas proporciona a los programadores una memoria virtual de gran tamaño a partir de una memoria física más pequeña.

- Sin embargo, hay que resolver dos problemas a la hora de implementar la paginación bajo demanda: hay que desarrollar un *algoritmo de asignación de marcos* y un *algoritmo de sustitución de páginas*.

Sustitución de páginas FIFO

El algoritmo más simple de sustitución de páginas es un algoritmo de tipo FIFO (first-in, first-out).

- El algoritmo FIFO asocia con cada página el instante en que dicha página fue cargada en memoria.
- Cuando hace falta sustituir una página, se elige la página más antigua.

Sustitución de páginas FIFO

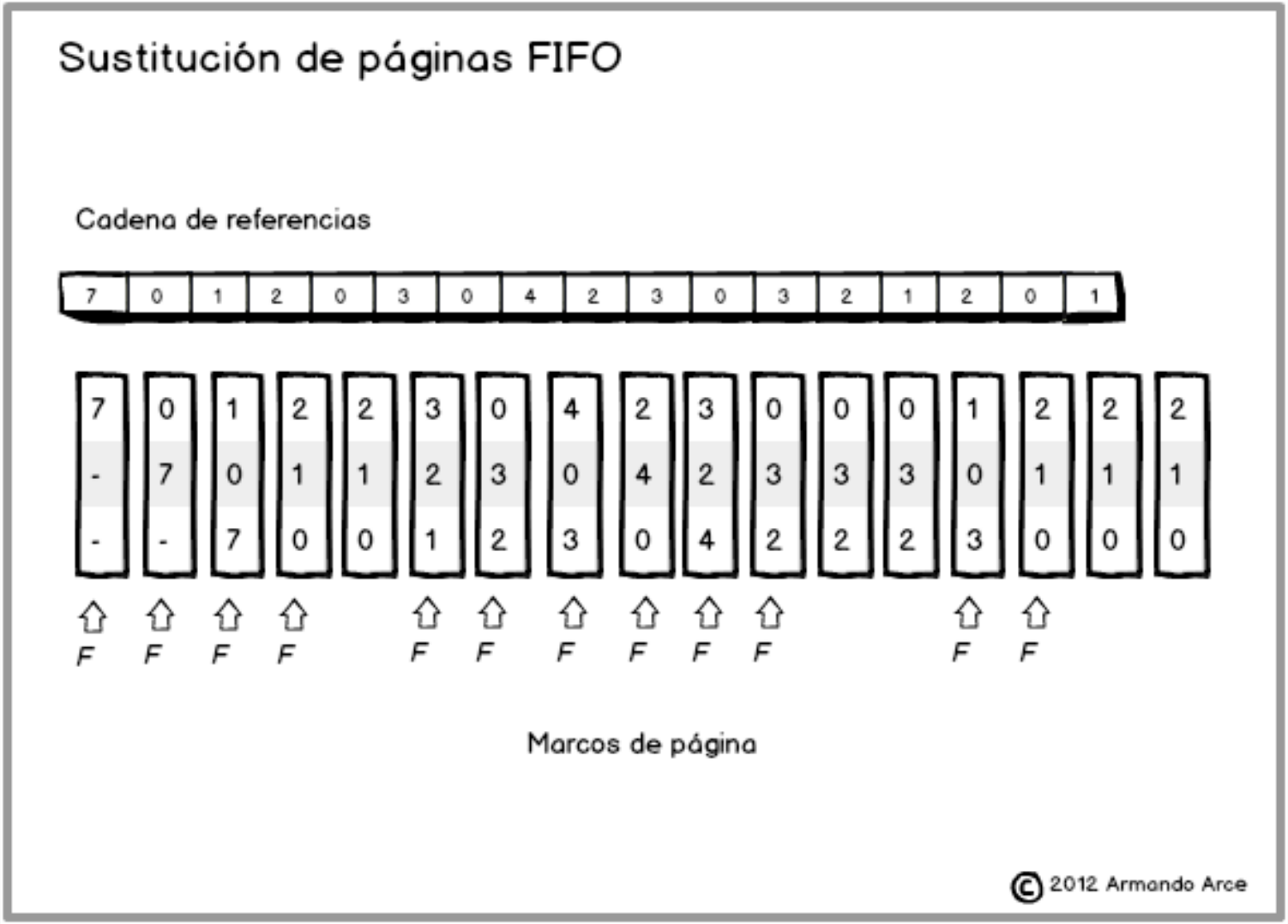

Figure 6:

Sustitución de páginas FIFO

El algoritmo FIFO es fácil de entender y programar, pero su rendimiento no siempre es bueno.

- Por ejemplo, la página sustituida podría ser una que contenga una variable que se utiliza constantemente.

Anomalía de Belady

La anomalía de Belady indica que para algunos algoritmos de sustitución de páginas, la tasa de fallos de página puede incrementarse a medida que se incrementa el número de marcos asignados.

- En particular, este problema se presenta en el algoritmo de sustitución FIFO.

Anomalía de Belady

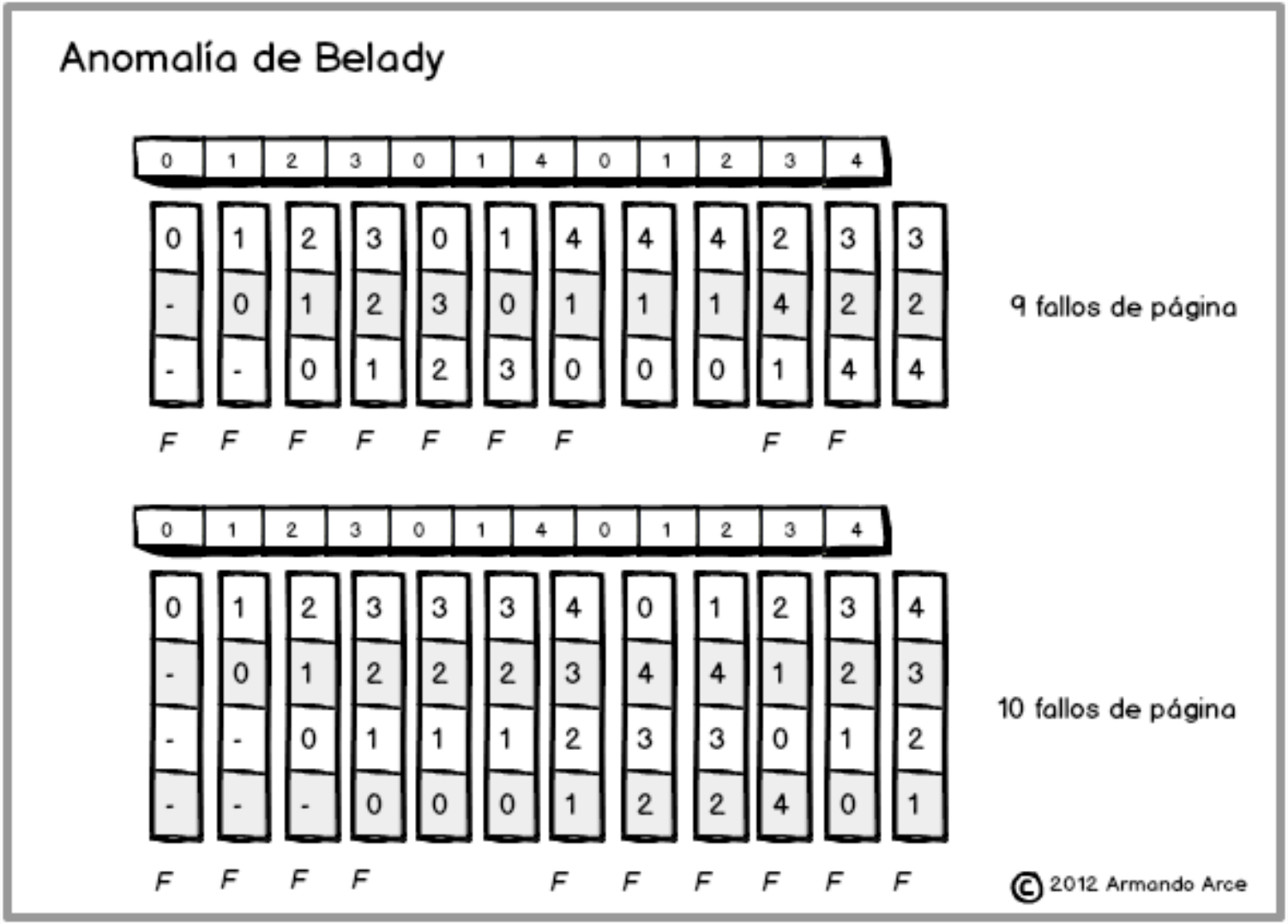

Figure 7:

Sustitución óptima de páginas

Un algoritmo óptimo de sustitución de páginas es aquel que tenga la tasa más baja de fallos de página de entre todos los algoritmos y que nunca esté sujeto a la anomalía de Belady.

- El algoritmo óptimo funciona de la siguiente manera: sustituir la página que no vaya a ser utilizada durante el período de tiempo más largo.
- La utilización de este algoritmo de sustitución de página garantiza la tasa de fallos de página más baja posible para un número fijo de marcos.

Sustitución óptima de páginas

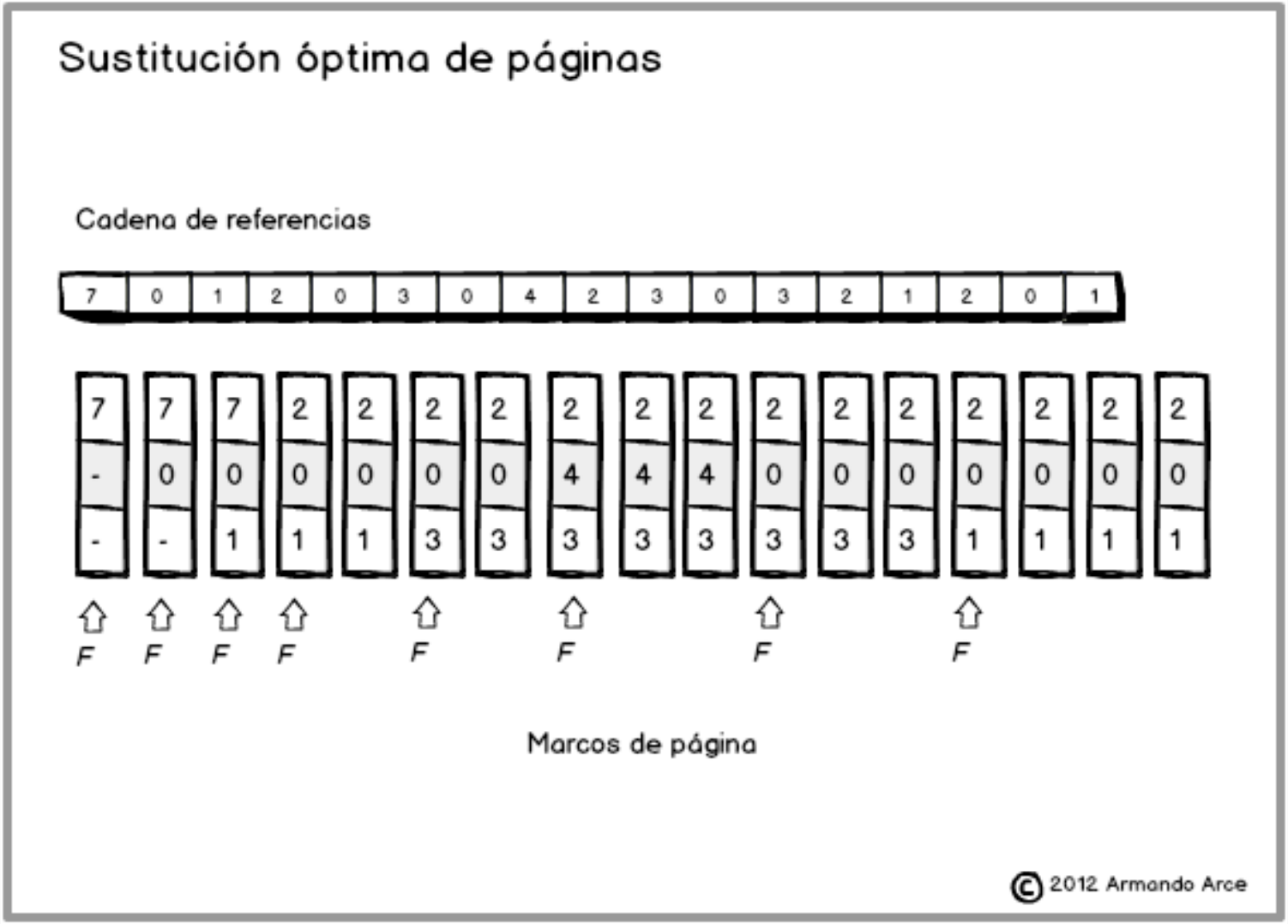

Figure 8:

Sustitución óptima de páginas

Desafortunadamente, el algoritmo óptimo resulta difícil de implementar, porque requiere un conocimiento futuro de la cadena de referencias.

- Como resultado, el algoritmo óptimo se utiliza principalmente con propósitos comparativos.

Sustitución de páginas LRU

El algoritmo de sustitución LRU asocia con cada página el instante correspondiente al último uso de dicha página.

- Cuando hay que sustituir una página, el algoritmo LRU selecciona la página que no haya sido utilizada durante un período de tiempo más largo.

Sustitución de páginas LRU

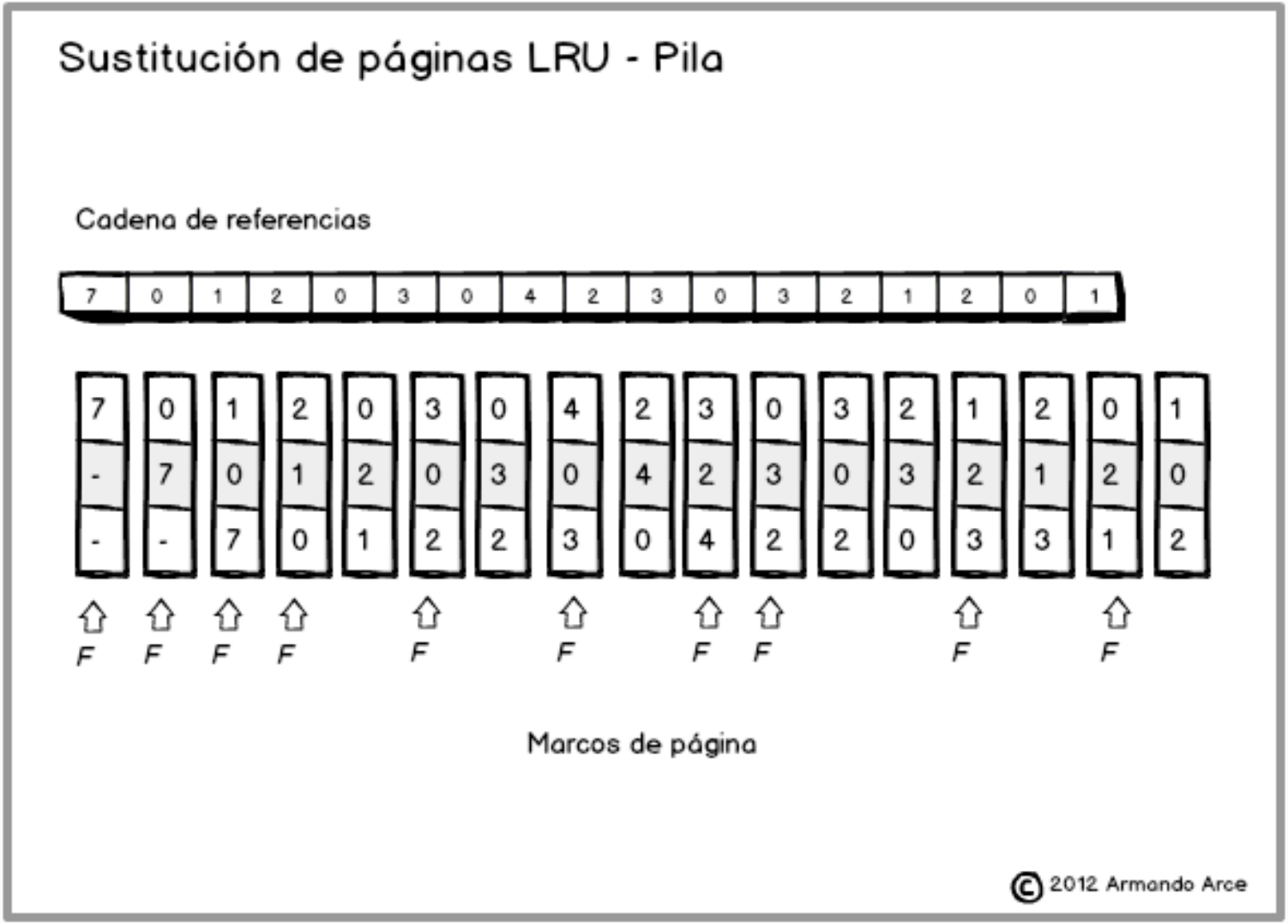

Figure 9:

Sustitución de páginas LRU

La política LRU es bastante buena.

- El principal problema es cómo implementar ese mecanismo de sustitución LRU. Un algoritmo LRU puede requerir una considerable asistencia hardware.
- El problema consiste en determinar un orden para los marcos definido por el instante correspondiente al último uso.
- Existen dos posibles implementaciones: Sustitución de páginas LRU y sustitución de páginas LRU

Sustitución de páginas LRU

- Contadores: Se asocia cada entrada en la tabla de páginas con un campo de tiempo de uso y se añade a la CPU un reloj lógico o contador. El reloj se incrementa en cada referencia a memoria.
- Cuando se realiza una referencia a una página, se copia el contenido del contador en el campo de tiempo de uso asociado a dicha página en la entrada de la tabla de páginas.
- De esta forma se sustituye la página que tenga el valor temporal menor. Este enfoque es adecuado para implementaciones mediante hardware.

Sustitución de páginas LRU

- Pila: Consiste en mantener una pila de números de página. Cada vez que se hace referencia a una página, se extrae esa página de la pila y se la coloca en la parte superior.
- De esta forma, la página más recientemente utilizada se encontrará siempre en la parte superior de la pila y la menos recientemente utilizada en la inferior.
- La mejor técnica para implementar este mecanismo es utilizar una lista doblemente enlazada. Este enfoque es adecuado para implementaciones mediante software.

Sustitución de páginas por aproximación LRU

Pocos sistemas proporcionan suficientes facilidades hardware como para implementar un verdadero algoritmo LRU.

- Sin embargo, en ocasiones se puede contar con un *bit de referencia* por página, que es activado por hardware cada vez que se hace referencia (leer o escribir) a esa página.
- Estos bits se encuentran en cada entrada de la tabla de páginas.

Uso de bits de referencia adicionales

Se puede disponer de información de ordenación adicional registrando los bits de referencia a intervalos regulares.

- Mediante un byte de 8 bits por cada página se puede hacer un corrimiento en cada intervalo e ingresar el nuevo bit de referencia por la izquierda del byte.
- De esta forma, estos registros de un byte contienen el historial de uso de las páginas de los últimos 8 períodos.
- Si se interpretan estos bytes como enteros sin signo, la página con el número más bajo será la menos recientemente utilizada y se podría sustituir.

Algoritmo de segunda oportunidad

El algoritmo de segunda oportunidad es igual a FIFO, pero cuando se selecciona una página, se inspecciona su bit de referencia.

- Si el bit es 0, se selecciona dicha página, pero si el bit es 1, se le da a la página una segunda oportunidad y se selecciona la siguiente página FIFO.
- Cuando se le da la segunda oportunidad a la página se debe borrar su bit de referencia. De esta forma una página que se utiliza suficientemente nunca será sustituida.

Algoritmo de segunda oportunidad

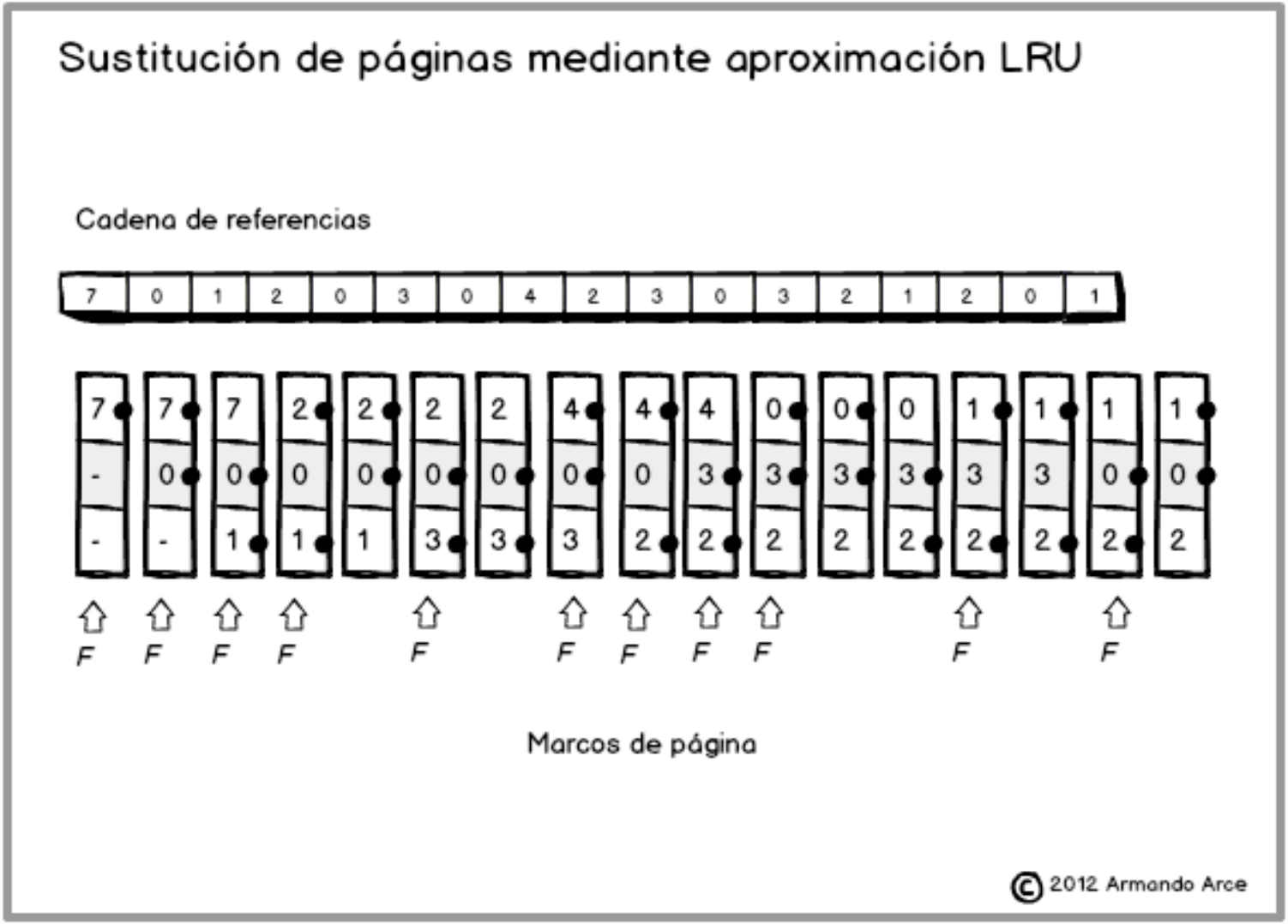

Figure 10:

Algoritmo de segunda oportunidad

Una forma de implementar este algoritmo (que se le conoce como el *algoritmo del reloj*) es un cola circular.

- Aquí se utiliza un puntero un puntero para indicar cuál es la siguiente página que hay que sustituir.
- Cuando hace falta un marco, el puntero avanza hasta que encuentra una página con un bit de referencia igual a 0.
- Al ir avanzando, va borrando los bits de referencia. Una vez que se encuentra una página víctima, ésta se sustituye y la nueva página se inserta en la cola en dicha posición.

Algoritmo mejorado de segunda oportunidad

Se puede mejorar el algoritmo de segunda oportunidad considerando el bit de referencia y un *bit adicional de modificación* (se activa sólo en escrituras) como un pareja ordenada. Con esto dos bits, se tienen cuatro casos posibles:

- (0,0) no se usado ni modificado recientemente: es la mejor página para sustitución
- (0,1) no se usando pero si se modificó: no es tan buena como la anterior
- (1,0) se ha usado pero está limpia: probablemente se vuelva a usar pronto
- (1,1) se ha usado y modificado; probablemente se vuela a usar y hay que escribir la página

Algoritmo mejorado de segunda oportunidad

Cada página pertenece a una de estas cuatro clases.

- Cuando hace falta sustituir una página, se utiliza el mismo esquema que en el algoritmo del reloj; pero se sustituye la primer página en la clase no vacío más baja.
- Por tanto, en este algoritmo se da preferencia a las páginas que no hayan sido modificadas.

Sustitución de páginas con contador

Hay otros algoritmos que pueden utilizarse para la sustitución de páginas. Por ejemplo, mantener un contador del número de referencias que se haya hecho a cada página y desarrollar los siguientes dos esquemas:

- El algoritmo de sustitución de páginas LFU (least frequently used, menos frecuentemente utilizada) requiere sustituir la página que tenga el valor más pequeño de contador. Sin embargo, puede surgir un problema cuando una página se utiliza con gran frecuencia durante la fase inicial de un proceso y luego ya no se la vuelve a utilizar.

Sustitución de páginas con contador

- El algoritmo de sustitución de páginas MFU (most frequently used, más frecuentemente utilizada) se basa en el argumento que la página que tenga el valor de contador más pequeño acaba probablemente de ser cargada y todavía tiene que ser utilizada.

Algoritmos de búfer de páginas

Además de un algoritmo de sustitución de páginas específico, a menudo se utilizan otros procedimientos.

- Por ejemplo, los sistemas suelen mantener un conjunto compartido de marcos libres.
- Cuando se produce un fallo de página, se selecciona un marco víctima como antes.

Algoritmos de búfer de páginas

Sin embargo, la página deseada se lee en un marco libre extraído de ese conjunto compartido, antes de escribir en disco la víctima.

- Esto permite que el proceso de reincide lo antes posible, sin tener que esperar a que se descargue la página víctima.
- Cuando la víctima se descarga por fin, su marco se añade al conjunto compartido de marcos libre.

Algoritmos de búfer de páginas

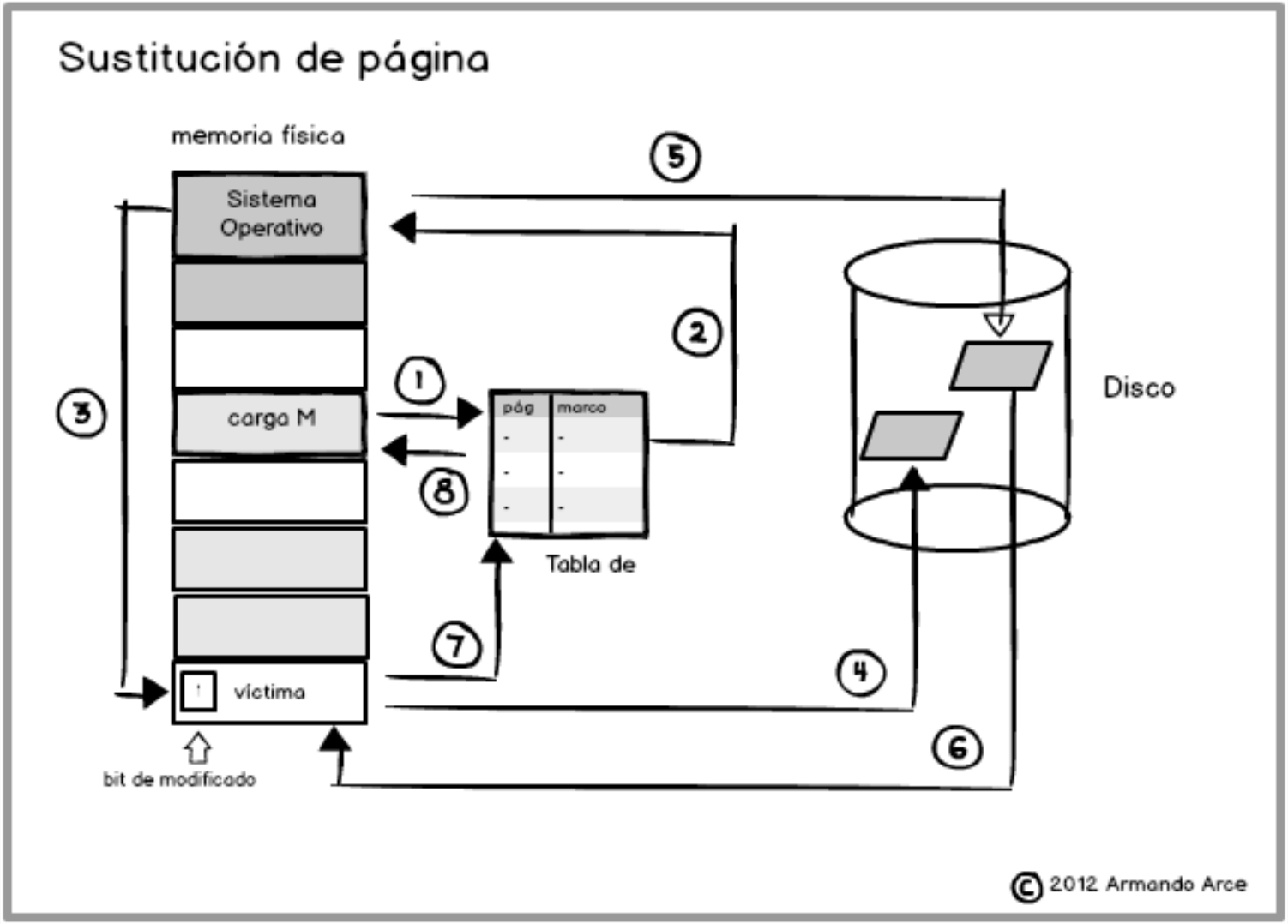

Figure 11:

Algoritmos de búfer de páginas

Una posible ampliación de este concepto consiste en mantener una lista de páginas modificadas.

- Cada vez que el dispositivo de paginación está inactivo, se selecciona una página modificada y se la escribe en el disco, desactivando a continuación su bit de modificación.
- Este esquema incrementa la probabilidad de que una página esté limpia en el momento de seleccionarla para sustitución, con lo que no será necesario descargarla.

Algoritmos de búfer de páginas

Otra posible modificación consiste en mantener un conjunto compartido de marcos libres, pero recordando qué página estaba almacenada en cada marco.

- Puesto que el contenido de un marco no se modifica después de escribir el marco en disco, la página antigua puede reutilizarse directamente a partir del compartido de marcos libres en caso de que fuera necesario y si dicho marco no ha dios todavía reutilizado.

Aplicaciones y sustitución de páginas

En ciertos casos, las aplicaciones que acceden a los datos a través de la memoria virtual del sistema operativo tienen un rendimiento peor que si el éste no proporcionara ningún mecanismo de búfer.

- Un ejemplo típico sería una base de datos, que proporciona sus propios mecanismos de administración de memoria y búferes de E/S.

Aplicaciones y sustitución de páginas

Debido a este problema, algunos sistemas operativos dan a ciertos programas especiales la capacidad de utilizar una partición del disco como si fuera una gran matriz secuencia de bloques lógicos, sin ningún tipo de estructura de datos propia de los sistemas de archivos.

- Esta matriz se denomina en ocasiones disco sin formato (disco crudo) y las operaciones de E/S que se efectúan sobre la matriz se denomina E/S sin formato (E/S cruda).
- La E/S sin formato no utiliza ninguno de los mecanismos del sistema de archivos: paginación, bloqueo de archivos, asignación de espacio, nombres de archivos y directorios.

Asignación de marcos

Se puede obligar al sistema operativo a asignar todo su espacio de búferes y de tablas a partir de la lista de marcos libres.

- Cuando este espacio no esté siendo utilizando por el sistema operativo, podrá emplearse para soportar las necesidades de paginación del usuario.
- También, se puede tratar de reservar en todo momento tres marcos libres dentro de la lista de marcos libres; así, cuando se produzca un fallo de página, siempre habrá un marco libre en el que cargar la página.
- Mientras que esté teniendo lugar el intercambio de la página, se puede elegir un sustituto, que será posteriormente escrito en disco mientras el proceso de usuario continúa ejecutándose.

Asignación de marcos

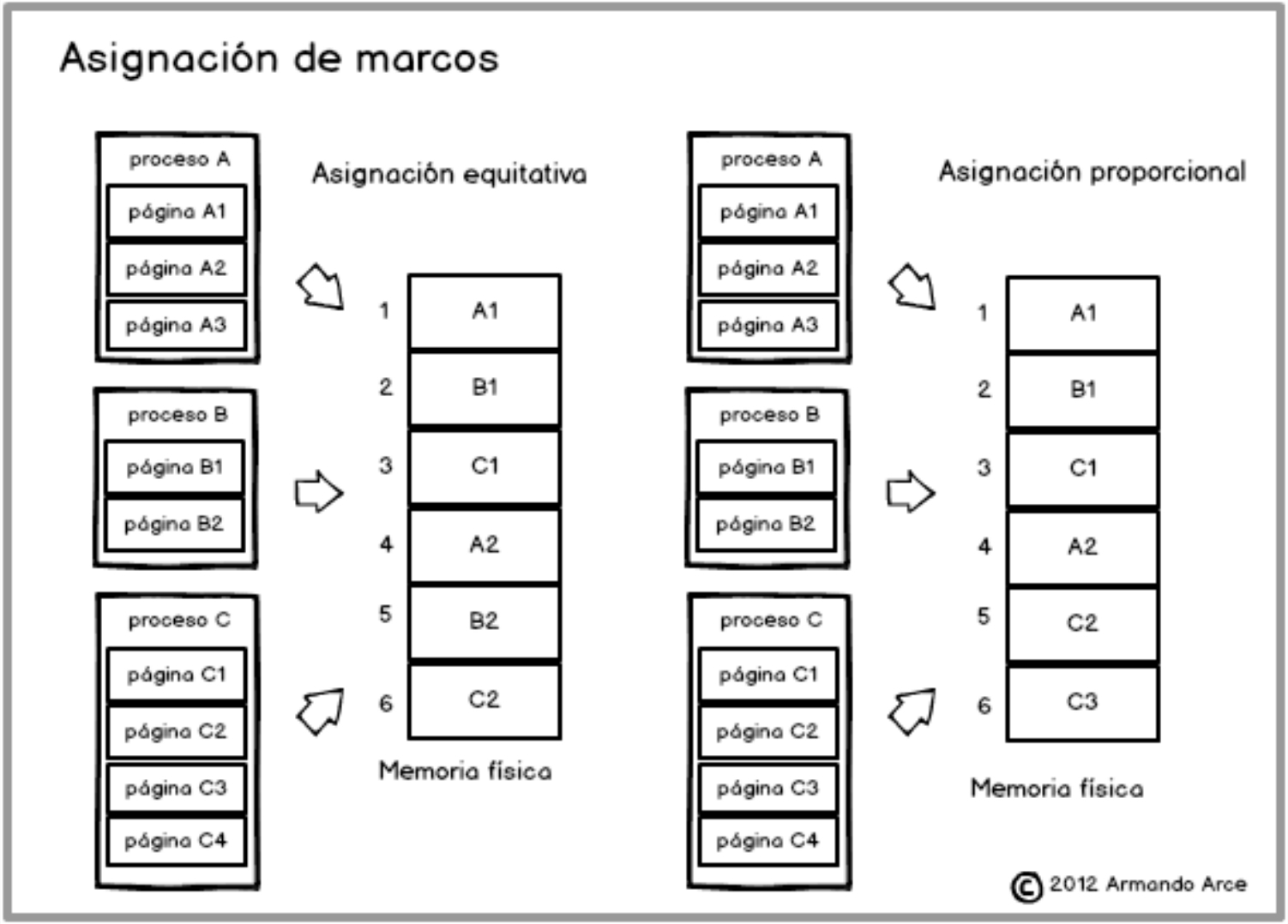

Figure 12:

Número mínimo de marcos

Es necesario establecer el número mínimo de marcos que será asignado a cada proceso por razones de rendimiento.

- Obviamente, a medida que el número de marcos asignados a un proceso se reduzca, se incrementará la tasa de fallos de páginas, ralentizando la ejecución del proceso.

Algoritmos de asignación

La forma más fácil de repartir m marcos entre n procesos consiste en dar a cada uno un número igual de marcos, m/n.

- Este sistema se denomina *asignación equitativa*. Una posible alternativa consiste en darse cuenta de que los diversos procesos necesitarán cantidades diferentes de memoria.
- Para resolver este problema, se puede utilizar una *asignación proporcional*, asignando la memoria disponible a cada proceso de acuerdo con el tamaño de éste.

Algoritmos de asignación

Tanto con la asignación equitativa como con la proporcional, las asignaciones concretas pueden variar de acuerdo con el nivel de multiprogramación.

- Si se incrementa el grado de multiprogramación, cada proceso perderá algunos marcos con el fin de proporcionar la memoria necesaria para el nuevo proceso.
- A la inversa, si se reduce el nivel de multiprogramación los marcos que hubieran sido asignados al proceso terminado pueden distribuirse entre los procesos restantes.

Algoritmos de asignación

Por su propia definición es conveniente asignar a los procesos de alta prioridad más memoria con el fin de acelerar su ejecución, en detrimento de los procesos de baja prioridad.

- Una solución consiste en utilizar un esquema de asignación proporcional en el que la cantidad de marcos dependa de las prioridades de los procesos.

Asignación global y local

Otro factor importante en la forma de asignar los marcos a los diversos procesos es el mecanismo de sustitución de página.

- Si hay múltiples procesos compitiendo por los marcos, se puede clasificar los algoritmos de sustitución de páginas en dos categorías amplias: *sustitución global* y *sustitución local*.
- La sustitución global permite a un proceso seleccionar un marco de sustitución de entre el conjunto de todos los marcos, incluso si dicho marco está asignado actualmente a algún otro proceso; es decir, un proceso puede quitar un marco a otro.
- El mecanismo de sustitución local requiere, por el contrario, que cada proceso sólo efectúe esa selección entre su propio conjunto de marcos asignado; es decir, el número de marcos asignados a un proceso no se modifica.

Asignación global y local

Un posible problema con el algoritmo de sustitución global es que un proceso no puede comprobar su propia tasa de fallos de página.

- El conjunto de páginas en memoria para un proceso dependerá no sólo del comportamiento de paginación de dicho proceso, sino también del de los demás procesos.
- Sin embargo, un algoritmo de sustitución local generalmente tiene un rendimiento menor debido a que un proceso no tiene a su disposición otras páginas de memoria menos utilizadas.

Archivos mapeados a memoria

En un lectura secuencia de un archivo de disco cada acceso al archivo requiere una llamada al sistema y un acceso al disco.

- En forma alternativa se pueden utilizar las técnicas de memoria virtual para tratar la E/S de un archivo como si fueran accesos rutinarios a memoria.
- Esta técnica, conocida con el nombre de *mapeo en memoria* de un archivo, permite asociar lógicamente con el archivo una parte del espacio virtual de direcciones.

Mecanismo básico

El *mapeo en memoria de un archivo* se lleva a cabo mapeando cada bloque de disco sobre una página (o páginas) de memoria.

- El acceso inicial al archivo se produce mediante los mecanismos ordinarios de paginación bajo demanda, provocando un fallo de página.
- Sin embargo, lo que se hace es leer una parte del archivo equivalente al tamaño de una página, extrayendo los datos del sistema de archivos y depositándolos en una página física.

Mecanismo básico

Las subsiguientes lecturas y escrituras en el archivo de gestionarán como accesos normales de memoria, simplificando así el acceso de archivo y también su utilización, al permitir que el sistema manipule los archivos a través de memoria en lugar de recurrir en la carga de trabajo adicional asociada a la utilización de las llamadas al sistema *read()* y *write()*.

Mecanismo básico

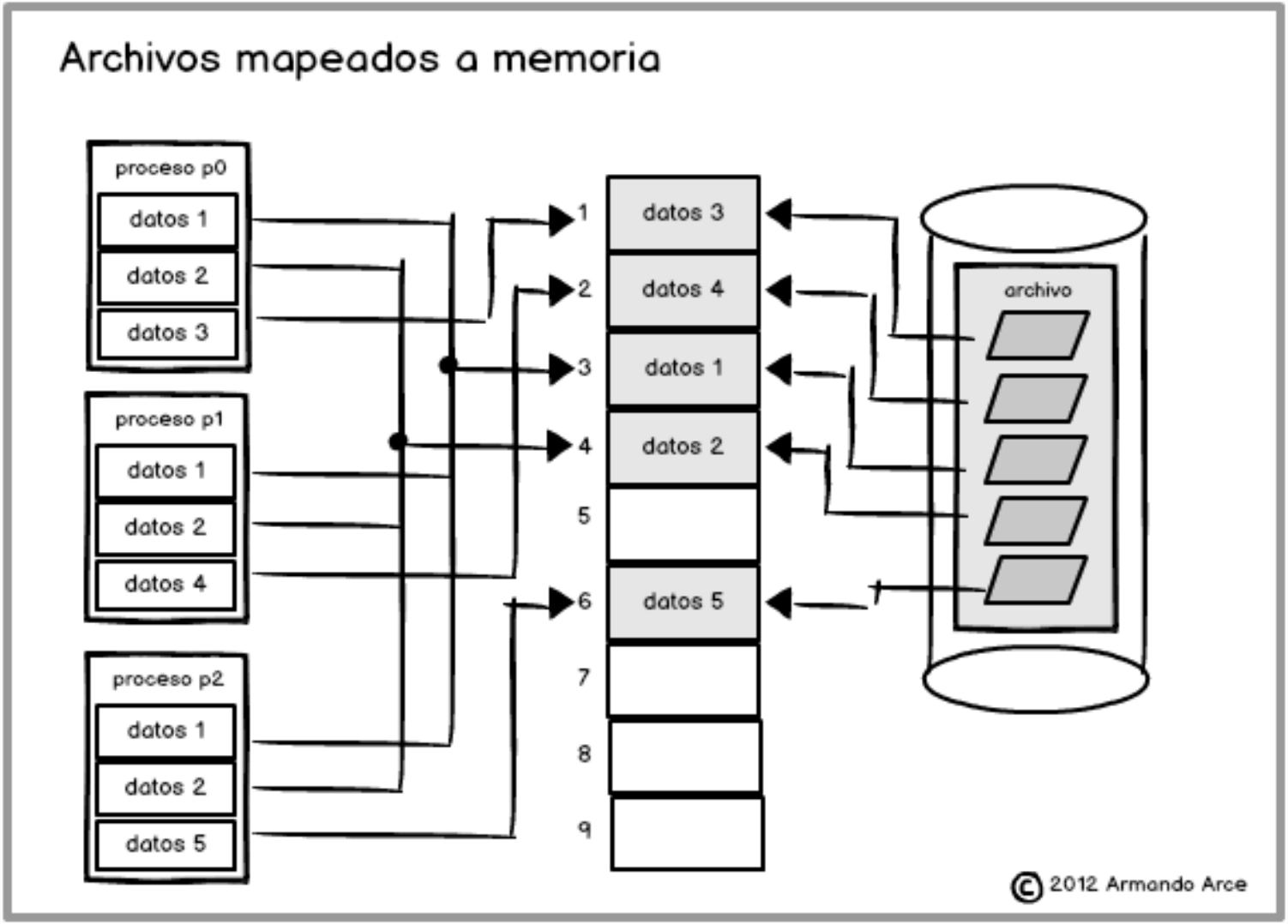

Figure 13:

Mecanismo básico

Algunos sistemas pueden actualizar el archivo físico hasta que se compruebe periódicamente si la página de memoria ha sido actualizada.

- Cuando el archivo se cierra, todos los datos mapeados se volverán a escribir en el disco y serán eliminados de la memoria virtual del proceso.
- También, algunos sistemas prefieren mapear en memoria un archivo independientemente de si ese archivo se ha especificado como archivo mapeado en memoria.

Mecanismo básico

Puede dejarse que múltiples procesos mapeen de forma concurrente el mismo archivo, con el fin de permitir la compartición de datos.

- Las escrituras realizadas por cualquiera de los procesos modificarán los datos contenidos en la memoria virtual y esas modificaciones podrán ser vistas por todos los demás procesos que hayan mapeado la misma sección del archivo.
- Las llamadas al sistema para mapeo en memoria pueden también soportar la funcionalidad de *copia durante la escritura*, permitiendo a los procesos compartir un archivo en modo de sólo lectura, pero disponiendo de sus propias copias de los datos que modifiquen.

E/S mapeada a memoria

Para permitir un acceso más cómodo a los dispositivos de E/S, muchas arquitecturas informáticas proporcionan *E/S mapeada a memoria*.

- En este caso, se reservan una serie de rangos de direcciones de memoria y se mapean esas direcciones sobre los registros de los dispositivos.
- Las lecturas y escrituras en estas direcciones de memoria hacen que se transfieran datos hacia y desde los registros de los dispositivos.
- Este método resulta apropiado para los dispositivos que tengan un rápido tiempo de respuesta, como los controladores de video.

Asignación de la memoria del kernel

La memoria del kernel suele asignarse a partir de un conjunto compartido de memoria compartida que es distinto de la lista utilizada para satisfacer las necesidades de los procesos normales que se ejecutan en modo usuario. Hay dos razones principales para que esto sea así:

- El kernel solicita memoria para estructuras de datos de tamaños variables, algunas de las cuales tienen un tamaño inferior a una página lo que puede provocar fragmentación.
- Ciertos dispositivos hardware interaccionan directamente con la memoria física y, consecuentemente, pueden requerir memoria que resida en páginas físicas contiguas.

Descomposición binaria

El sistema de *descomposición binaria* (buddy system) asigna la memoria a partir de un segmento de tamaño fijo compuesto de páginas físicamente contiguas.

- La memoria se asigna a partir de este segmento mediante un *asignador de potencias de 2*, que satisface las solicitudes en unidades cuyo tamaño es una potencia de 2 (4KB, 8KB, 16KB, etc.)
- Toda solicitud que no tenga el tamaño apropiado se redondeará hasta la siguiente potencia de 2 más alta.

Descomposición binaria

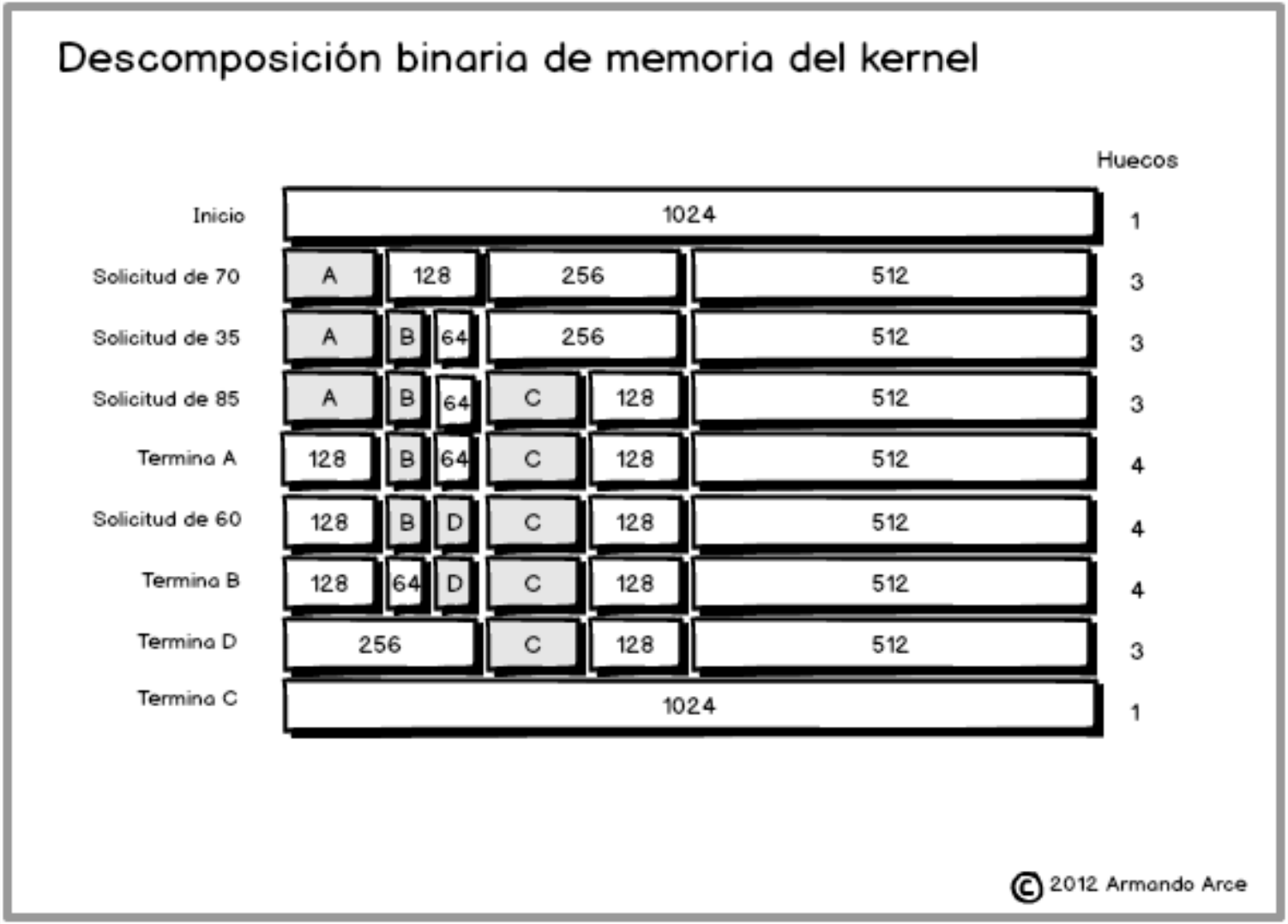

Figure 14:

Descomposición binaria

Si, por ejemplo, el segmento de memoria inicial es de 256 KB y se solicitan 21 KB de memoria.

- El segmento se dividirá inicialmente en dos subsegmentos cada uno de los cuales tendrá un tamaño igual a 128KB.
- Uno de estos subsegmentos se volverá luego a dividir en dos subsegmentos de 64 KB, y así sucesivamente conforme que requieran más segmentos.

Descomposición binaria

Una ventaja del sistema de descomposición binaria es la rapidez con la que pueden combinarse subsegmentos adyacentes para formar segmentos de mayor tamaño, utilizando la técnica de *consolidación* entre segmentos que forman pareja.

- La desventaja obvia del sistema de descomposición binaria es que el redondeo de la siguiente potencia de 2 más alta producirá, muy probablemente, que aparezca fragmentación dentro de los segmentos asignados.

Asignación de franjas

Una segunda estrategia para la asignación de la memoria del kernel se conoce con el nombre de asignación de franjas (slabs).

- Una franja está formada por una o más páginas físicamente contiguas.
- Una caché está compuesta de una o más franjas.

Asignación de franjas

Existe una única caché por cada estructura de datos del kernel distinta; por ejemplo, hay una caché para la estructura de datos que representa los descriptores de los procesos, otra caché distinta para los objetos de archivo, otra más para los semáforos, etc.

- Cada caché se rellena de objetos que sean instancias de la estructura de datos del kernel que esa caché representa.

Asignación de franjas

Cuando se crea una caché, se asigna a la caché un cierto número de objetos (que están inicialmente marcados como libres).

- El número de objetos de la caché dependerá del tamaño de la franja asociada.
- Cuando hace falta un nuevo objeto para una estructura de datos del kernel, el asignador puede asignar cualquier objeto libre de la cache.
- El objeto asignado se marca como usado.

Asignación de franjas

Una ventaja de la asignación de franjas es que no se pierde memoria debido a la fragmentación, pues las estructuras de datos en una misma franja son del mismo tipo y generalmente del mismo tamaño.

- Otra ventaja es que las solicitudes de memoria se satisfacen rápidamente, pues los objetos están creados de antemano.

Referencia bibliográfica

A.SILBERSCHATZ, P. GALVIN, y G. GAGNE, Operating Systems Concepts, Cap.9, 9a Edición, John Wiley, 2013.