8.25

Examen parcial 2
Sistemas Operativos II

18/06/2015

Nombre, Apellido y Legajo: _____

1. La figura 1 representa el estado actual de un sistema de memoria manejado con el Buddy System. Puede verse que hay dos bloques de tamaño 128 bytes, uno de 512 bytes y ninguno de tamaño 256. Se muestra también la ubicación de cada bloque libre en la memoria RAM.

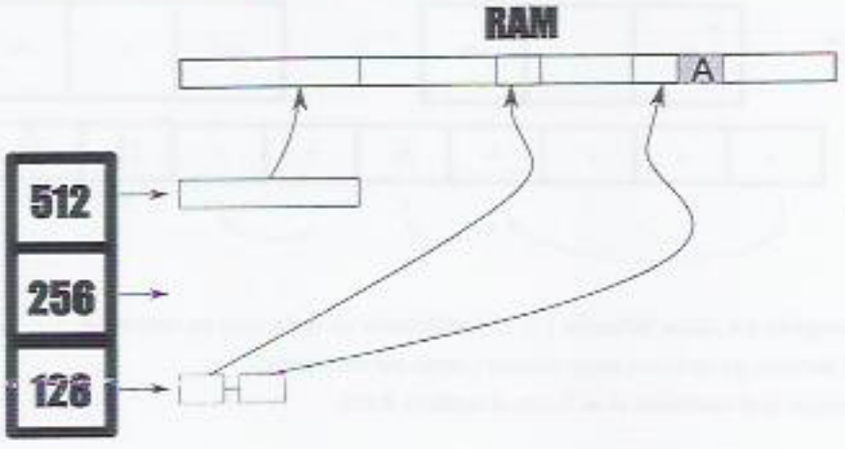


Figura 1: Estado actual del sistema Buddy

   (a) Muestre el estado resultante si se pide un bloque de tamaño 256.
   (b) Muestre el estado resultante si se libera el bloque A de tamaño 128.

2. Indique cuál es el principal problema que aparece cuando se utiliza segmentación y cómo se hace para solucionarlo.

3. Indique verdadero o falso justificando su respuesta.
   - A. LRU tiene mejor rendimiento que FIFO en todos los casos. F
   - B. El algoritmo óptimo tiene mejor rendimiento que LRU en todos los casos. F
   - C. Si un proceso Linux hace `fork` siempre se crea una nueva tabla de paginación. V
   - D. Si un proceso Linux hace `fork` siempre se crea una copia de todas sus páginas en memoria. F
   - E. El SO es quien prende el bit referenced ante cada acceso a memoria. F
   - F. El hardware apaga el bit de referenced periódicamente. F

4. En qué consiste y para qué se utiliza reverse mapping en GNU/Linux.

5. Indique qué tipo de filesystem de los vistos en clase se ajusta mejor para las siguientes situaciones, justificando brevemente su respuesta.

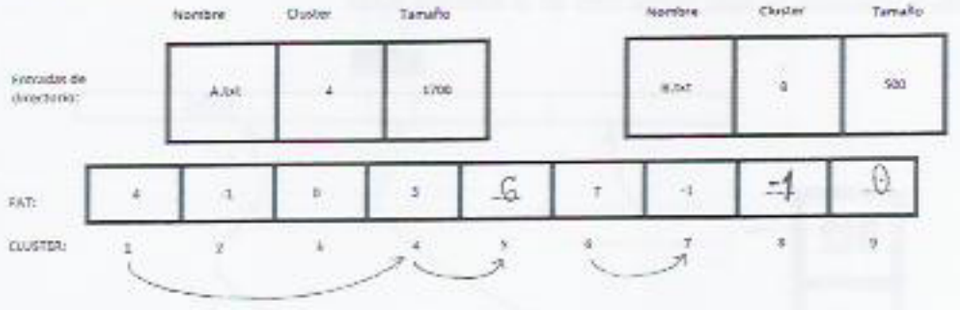
   A. Se requieren almacenar reclamos para archivarlos (cada reclamo se guarda en un archivo separado). Un reclamo nunca se modifica ni se borra.

   B. IDEM anterior, pero ahora cada archivo puede crecer (los datos anteriores no se alteran).

   C. Cada archivo puede tener más de un nombre.

   D. No se quiere mantener una estructura de datos separada para gestionar el espacio libre.

6. Se tiene un disco con 9 clusters de 512 bytes utilizando un sistema basado en FAT con el estado (muy simplificado) que se muestra en la figura. Suponga que el disco sólo contiene bloques de datos. -1 indica que no hay más clusters en esta lista y 0 indica que el bloque está libre.



   (a) Complete los datos faltantes (_ _ _) justificando en cada caso su respuesta.

   (b) El sistema de archivos tiene errores ¿cómo los corregiría?.

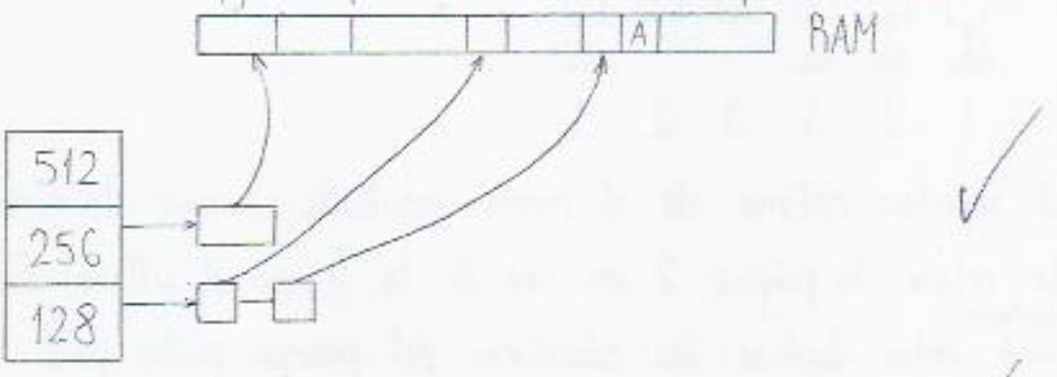
   (c) Indique qué cambiará si se borra el archivo A.txt.

Parcial de Sistemas Operativos II

1/2
jue. 18/06/2015

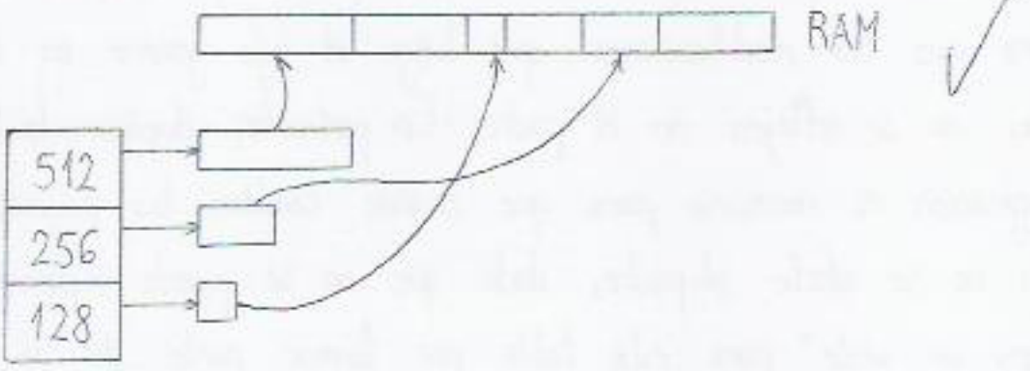
Segundo parcial de Sistemas Operativos II

1.a.



✓

1.b.



✓

Nota: se efectúa la transición desde el estado inicial, no desde el estado obtenido en (1.a).

3.A. Falso. Si se da una traza de accesos a páginas tal que siempre la página que más vaya a tardar en usarse sea la que lleve más tiempo desde su última carga en RAM, entonces FIFO y LRU determinarán ambos la solución óptima, pero FIFO tendrá menor carga administrativa.

3.B. Hay casos en que el algoritmo óptimo y LRU tienen el mismo rendimiento (considerando rendimiento solo como la cantidad de fallos generados de página, ya que la sobrecarga administrativa no aplica al ser inimplementable el algoritmo óptimo). Puntualmente, son las trazas de la forma casos tales:

$$\langle x_1, x_2 \dots x_n, x_n x_{n-1} \dots x_1 \rangle$$

Ejemplo: en un sistema con capacidad para 2 páginas.

⟨1,2,3,3,2,1⟩

El algoritmo óptimo da el mismo resultado, aunque también admite retirar la página 2 en vez de la 3 en el último fallo.

Verdadero.

3.C. fork debe duplicar las estructuras del proceso padre para el hijo, para que las modificaciones que luego el hijo genere en las mismas no se reflejen en el padre. En particular, duplicar la tabla de paginación es necesario para que cuando cambien las páginas del hijo no se afecte al padre, dado que no se puede implementar "copy on write" para esta tabla por formar parte de la memoria del núcleo. De la misma forma también se busca evitar que el padre afecte al hijo ✓

3.D. Falso. En la medida de lo posible, Linux evita copiar páginas. Usa "copy on write", es decir, mantiene una única "instancia" mientras ambos procesos solo lean la página en cuestión, y recién la duplica cuando un de los procesos la escriba. ✓

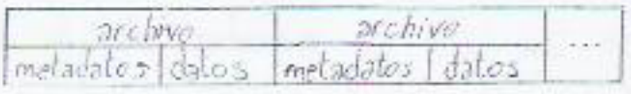
3.E. Falso. De eso debe encargarse el hardware. El sistema operativo no interviene en los accesos a memoria de los procesos salvo que haya una excepción. ✓

3.F. Falso. El sistema operativo se encarga de esa. ✓

4. En Linux, "reverse mapping" se refiere a la localización de una página a partir de su marco asociado. Aparte de mantener una tabla de paginación por proceso, que asocia páginas a marcos, Linux

mantiene una tabla global que hace la asociación inversa: indica, para cada marco, qué página tiene asociada si tiene alguna, y también otra información como si está ~~efectivamente~~ su página guardada en disco (en el área de intercambio), si se puede enviar a disco para liberar el marco o necesariamente hay que mantenerla en memoria porque se está efectuando alguna operación de entrada/salida sobre ese espacio de memoria y demás.
Un uso fundamental de "reverse mapping" es encontrar una página víctima cuando se quiere liberar un marco V

5.A. Un sistema de archivos que particione el espacio disponible de forma que cada archivo quede almacenado en una única partición, de forma secuencial e ininterrumpida, incluyendo en la misma tanto datos como metadatos. Similar a las primeras versiones de TAR, y tal vez a ISO-9660 o UDF.



Esto optimiza el uso del espacio, no hay fragmentación ni interna ni externa. Agregar un archivo requiere solo escribir al final. Podría implementarse alguna estructura de directorio para evitar recorrer todo el espacio al buscar un archivo. V

5.C. Un sistema de archivos que guarde por separados los nombres del resto de metadatos ~~(por ejemplo~~ en estructuras como i-nodos, de modo que se pueda asociar varios nombres a la misma estructura. Los sistemas de archivos de sistemas Unix implementan esto.

5.D. Un sistema de archivos FAT. La FAT es una tabla que sirve tanto para gestionar el espacio libre como para gestionar el espacio ocupado manteniendo listas enlazadas que determinan todos los clústeres que están asignados a cada archivo.

5.B. Un sistema similar al de (5A) pero que permita indicar en la cabecera de cada archivo si hay alguna partición adicional que debe considerarse la continuación de la partición actual. Esta es una de las soluciones más simples, se genera fragmentación externa pero las escrituras son eficientes.

6.a. En 8 debe ir -1 porque B.txt ocupa menos de 1 clúster.
En 5 debe ir el número de un clúster que luego lleve a uno con -1.
Queda un clúster libre, en el que va 0.

b. Poniendo en 0 los clústeres a los que no se llega en la lista de A.txt ni la de B.txt.

c. Se altera el primer byte de la entrada de directorio de A.txt y se ponen en 0 las entradas de la FAT correspondientes a clústeres de A.txt.