Respuestas del test

Nº	a	b
1	X	
2	X	
3		X
4	X	
5	X	
6		X
7		X
8	X	
9		X
10		X
11		X
12	X	
13	X	
14	X	
15		X
16		X
17		X
18	X	
19	X	
20		X

Nº	a	b
21	X	
22	X	
23	X	
24		X
25	X	
26	X	
27		X
28	X	
29		X
30	X	
31	X	
32		X
33		X
34	X	
35	X	
36		X
37		X
38		X
39	X	
40		X

Introducción a los Sistemas Operativos

Examen de teoría: problemas

17 de mayo de 2023

Apellidos:	Nombre:
Grupo: ☐ 1 ☐ 2 ☐ 3 ☐ PCEO	DNI:

Instrucciones:

- Todas las respuestas deben escribirse con **bolígrafo**.
- Se valorará la **exactitud, completitud y brevedad** de todas las respuestas, que deberán ser **RAZONADAS**. No tendrán validez las respuestas no justificadas.
- Esta parte vale 3 puntos y, por tanto, representa el 30 % de la nota del examen final de teoría.

1. (1,5 puntos) Tenemos un sistema de ficheros UNIX estándar con nodo-i de 64 bytes, 10 entradas directas, BSI, BDI y BTI. Tamaño de bloque 1024 bytes y tamaño del número de bloque de 4 bytes. El SO mantiene una caché de bloques suficientemente grande.

   Queremos leer 5 000 KB ($5\,000 \cdot 2^{10}$ bytes) a partir del byte 201 533 000, inclusive (recuerda que comenzamos a numerar los bytes de un fichero desde 0). Se pide responder a las siguientes cuestiones:

   a) (0,2) ¿Qué cantidad de bloques de datos tendremos que leer del fichero?

   b) (0,8) ¿Qué rutas hay que seguir desde del nodo-i para obtener los bloques de disco correspondientes al primer y último bloque de datos respectivamente que se pide leer. Un ejemplo de respuesta para un bloque sería: en el BDI independiente se accede al BSI 3, y dentro de ese BSI, se accede a la entrada 25, donde estará el número de bloque. La explicación debe ser razonada.

   c) (0,25) ¿Cuántos bloques BSI, BDI y BTI hemos tenido que leer de disco para leer secuencialmente todos los bloques de datos que nos piden?

   d) (0,25) Si el sistema de ficheros hubiese sido implementado por lista ligada sin índice, ¿cuántos bloques habríamos tenido que leer?

2. (1,5 puntos) Tenemos un computador con 512KiB de RAM, memoria virtual paginada con direcciones virtuales de 24 bits y físicas de 20, tamaño de página de 4KiB. Responda a las siguientes cuestiones:

   a) (0,1) Número de páginas físicas que tiene el sistema.

   b) (0,1) Número de entradas que tendría una tabla de traducción de páginas lineal.

   c) (0,1) Número de entradas que tendría como mínimo la tabla de traducción de una tabla de páginas invertida en este sistema.

   d) (0,2) ¿Para qué podríamos querer que una tabla de páginas invertida tuviera más entradas que las anteriores? Razónalo brevemente.

   e) (0,5) Dados los siguientes accesos a memoria de un mismo proceso, y teniendo en cuenta que estamos utilizando un algoritmo de reemplazo NRU, **dibuja la evolución en cada acceso** del contenido de los 4 marcos asignados al proceso (marco0-marco3). En caso de empate, tanto para el reemplazo como para la asignación de marcos, se seleccionará el marco con el número en su nombre más pequeño de los 4 disponibles. **Usa para responder a esta parte la tabla que hay en la otra cara del enunciado**.

   f) (0,5) Teniendo en cuenta que estamos utilizando una **tabla de página de dos niveles**, con 64 entradas cada uno, dibuja cómo queda la tabla de traducción tras los accesos del apartado anterior. Deja claro cuáles son los índices de cada nivel que se están usando e indica en el primer nivel de la tabla el valor de los bits de validez, y en el segundo nivel, los bits de validez junto con los valores de todos los campos que consideres que debe tener una entrada de la tabla en este contexto.

Orden	DV	L/E	NPV	A/F	Marco0 | Bits	Marco 1 | Bits	Marco 2 | Bits	Marco 3| Bits
1	0x01F358	E						
2	0x3E1200	E						
3	0x0C54FC	L						
4	0x088CCC	L						
TIC								
5	0x0C5230	L						
6	0x048ABC	E						
7	0x01F444	L						
8	0x088888	E						
9	0x3E1440	L						
TIC								
10	0x0C527E	L						

Soluciones

1. La solución de los diferentes apartados sería la siguiente:

   a) Para saber el número de bloques que leeremos, tendremos que saber en qué bloques comienzan el primer byte a leer y último, ya que puede ser que la lectura comience por mitad de un bloque. Para ello dividimos el número de byte entre el tamaño de bloque, y obtenemos: $201533000/1024 = 196809{,}57$ y $(201533000 + 5000 * 1024 - 1) = 206652999$, que si lo dividimos entre 1024 nos da $201809{,}57$. Por tanto tendremos que leer desde el bloque 196809 al 201809 ambos inclusive, dando un total de 5001 bloques.

   b) Para localizar la entrada dentro del nodo-i correspondiente a un bloque concreto, por ejemplo el 196809, tendremos que restarle 10 correspondientes a las entradas que se encuentran en el propio nodo-i, dando 196799. Ahora dividimos entre las 256 entradas que tiene cada BSI y nos dirá en qué BSI se encuentra el número de bloque: $196799/256 = 768{,}75$, es decir, el 768 contando desde 0 y desde el BSI independiente. Para saber en qué entrada de ese BSI estará el dato del bloque que buscamos, haremos el módulo: $196799 \bmod 256 = 191$.

   Puesto que se encuentra en el BSI 768, no está en el BSI independiente (sería el BSI 0), así que tendremos que buscar dónde se halla ese BSI. Para ello, restaremos 1 por el BSI independiente, y dividiremos el número entre el número de BSIs que contiene un BDI. Esto nos dará el número de BDI donde se encuentra: $(768 - 1)/256 = 767/256 = 2{,}996$. Así que está en el BDI 2. Con el módulo sabremos en qué entrada de ese BDI se encuentra: $767 \bmod 256 = 255$. Por tanto ya sabemos que el BSI buscado se encuentra en la entrada 255 del BDI 2. Como el BDI independiente es el BDI 0, el 2 se encontrará en el BTI, así que ya sabemos que estará en el BDI $2 - 1 = 1$ contando desde 0 en el BTI.

   Resumiendo, el número del bloque de disco correspondiente al bloque de datos del fichero número 196809 se encuentra en el BDI 1 del BTI, BSI 255 de ese BDI, y entrada 191 de ese BSI.

   Haciendo las mismas operaciones con el bloque final, obtendremos:

   - $(201809 - 10)/256 = 201799/256 = 788{,}28$, luego el BSI es el 788 contando desde 0, y la entrada en el BSI sería la $201799 \,\%256 = 71$
   - calculamos el BDI dividiendo el número de BSI-1 entre 256 BSIs que tiene un BDI, $(788 - 1)/256 = 787/256 = 3{,}074$, lo que nos da el BDI 3, y dentro de él sería el BSI $787 \bmod 256 = 19$.
   - Al ser un BDI mayor que 0, se encontrará en el BTI, en su entrada $3 - 1 = 2$.

   Así que vemos que el bloque de datos estará en la entrada 71 del BSI que se encuentra en la entrada 19 del BDI que se se encuentra en la entrada 2 del BTI.

   c) Como sabemos dónde se encuentran el primer número de bloque en el nodo-i y el último, podemos examinar la figura 1 y contar el número de bloques que tendremos que leer.

   Leeríamos el bloque BTI, después el BDI número 1, y el BSI número 255, donde encontraríamos la entrada 191 que corresponde al primer bloque de datos. Seguiríamos leyendo entradas hasta que se agotara el BSI. Para continuar tendríamos que localizar el siguiente, que estaría en la siguiente entrada BTI, es decir, el BDI 2, que habría que leer, seguido del BSI 0 de ese BDI. Seguiríamos leyendo los BSIs 1, 2 y sucesivos conforme avanzamos por los datos, hasta llegar al BSI 19 (como empiezan a contar desde 0, serían 20 BSIs leídos en ese BDI), donde ya encontraríamos el resto de las entradas de bloques de datos que faltan. Resumiendo: BTI + BDI 1 + BSI 255 + BDI 2 + BSI 0 + BSI 1 + . . . + BSI 19 = 24 bloques de metadatos.

   d) Al ser un sistema de ficheros con lista ligada, cada bloque tiene un puntero al siguiente, así que contendrá 1020 bytes realmente. Además para llegar al primer bloque a leer hay que recorrer todos los anteriores desde el principio, así que solamente tendremos que calcular la dirección del último bloque.

   Por tanto, para saber el número de accesos que necesitamos hacer, dividiremos 206652999 entre 1020, dando $206652999/1020 = 202600{,}98$, es decir, que tendremos que leer hasta el bloque 202600 (contando desde el 0), lo que supone leer 202601 bloques en total.

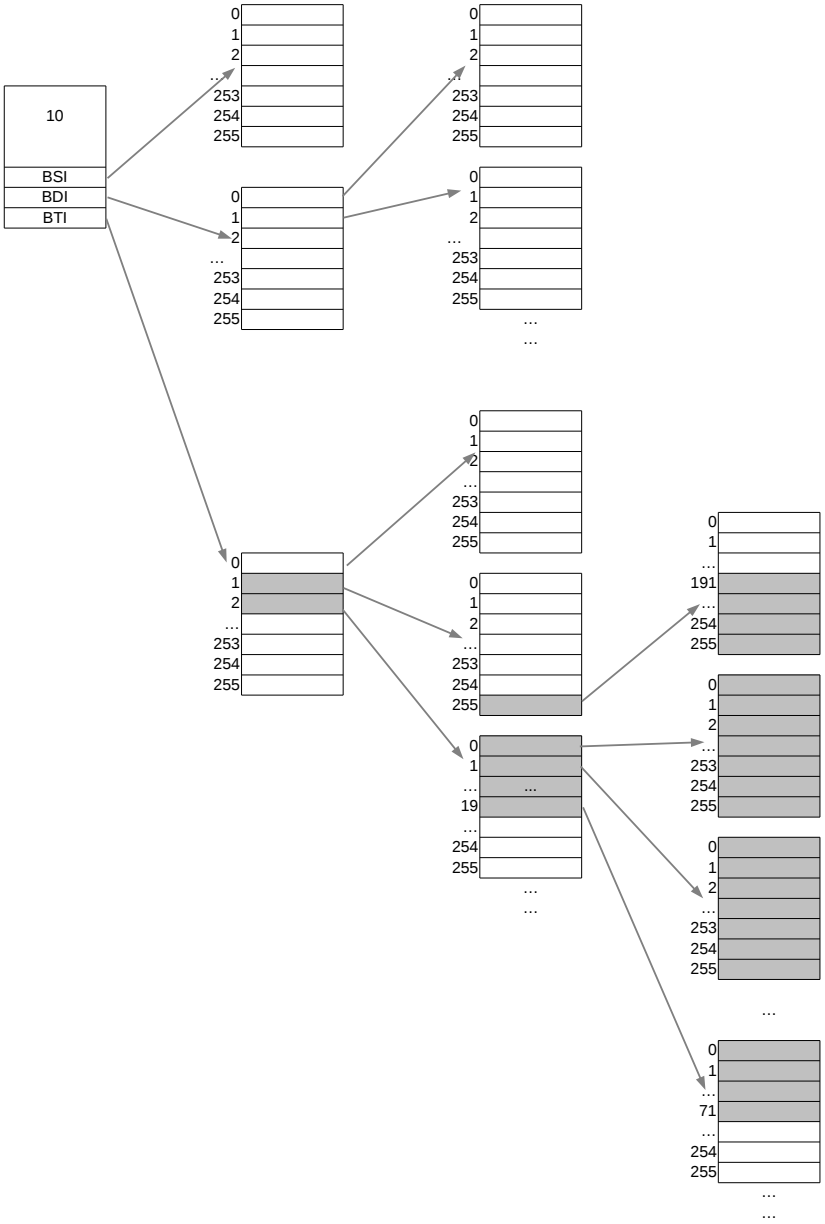

Figura 1: Nodo-i con los bloques de metadatos. Las entradas pertenecientes a los bloques del fichero que se van a leer están sombreadas.

2. La solución de cada apartado se muestra a continuación:

   a) El sistema tendría $512 \cdot 1024/4096 = 128$ páginas físicas. Potencialmente podría llegar a tener hasta 256 ($2^{20}/2^{12} = 2^8 = 256$).

   b) El número de entradas de la tabla de páginas lineal sería $2^{24}/2^{12} = 2^{12} = 4096$.

   c) Una tabla de páginas invertida tiene tantas entradas como marcos físicos tiene el sistema. En este caso tendría 128 entradas.

   d) Un posible motivo sería permitir con las entradas adicionales la compartición de páginas entre procesos. Puesto que sólo hay una tabla de páginas invertida, si queremos que dos procesos tengan compartido código —por ejemplo—, tendremos que tener algún sitio donde guardar las dos traducciones, y una posibilidad sería tener más entradas disponibles en la tabla.

   e) Las direcciones virtuales de memoria tienen que dividirse en página virtual y desplazamiento, ya que lo que nos interesa es la página virtual. Al tener una página de 4096 bytes, la separación es muy fácil, ya que simplemente tendremos que quitar los tres últimos dígitos hexadecimales de la dirección (12 bits). El resto será la página virtual, que es lo que usamos en el cuadro 1, donde se describe qué hay en cada marco físico tras cada acceso.

   f) La figura 2 muestra la tabla de páginas resultante.

Orden	DV	L/E	NPV	PT1/PT2	A/F	Marco0 | Bits	Marco 1 | Bits	Marco 2 | Bits	Marco 3| Bits
1	0x01F358	E	0x01F	0/31	F	**0x01F | RM**			
2	0x3E1200	E	0x3E1	15/33	F	0x01F | RM	**0x3E1 | RM**		
3	0x0C54FC	L	0x0C5	3/5	F	0x01F | RM	0x3E1 | RM	**0x0C5 | R**	
4	0x088CCC	L	0x088	2/8	F	0x01F | RM	0x3E1 | RM	0x0C5 | R	**0x088 | R**
TIC						**0x01F | M**	**0x3E1 | M**	**0x0C5 |**	**0x088 |**
5	0x0C5230	L	0x0C5	3/5	A	0x01F | M	0x3E1 | M	**0x0C5 | R**	0x088 |
6	0x048ABC	E	0x048	1/8	F	0x01F | M	0x3E1 | M	0x0C5 | R	**0x048 | RM**
7	0x01F444	L	0x01F	0/31	A	**0x01F | RM**	0x3E1 | M	0x0C5 | R	0x048 | RM
8	0x088888	E	0x088	2/8	F	0x01F | RM	**0x088 | RM**	0x0C5 | R	0x048 | RM
9	0x3E1440	L	0x3E1	15/33	F	0x01F | RM	0x088 | RM	**0x3E1 | R**	0x048 | RM
TIC						**0x01F | M**	**0x088 | M**	**0x3E1 |**	**0x048 | M**
10	0x0C527E	L	0x0C5	3/5	F	0x01F | M	0x088 | M	**0x0C5 | R**	0x048 | M

Cuadro 1: Tabla con el contenido de los marcos tras cada acceso. El cuadro muestra el valor de página virtual, si se produce acierto o fallo de página y el contenido de los 4 marcos de página con sus bits de control (RM). En negrita los cambios provocados.

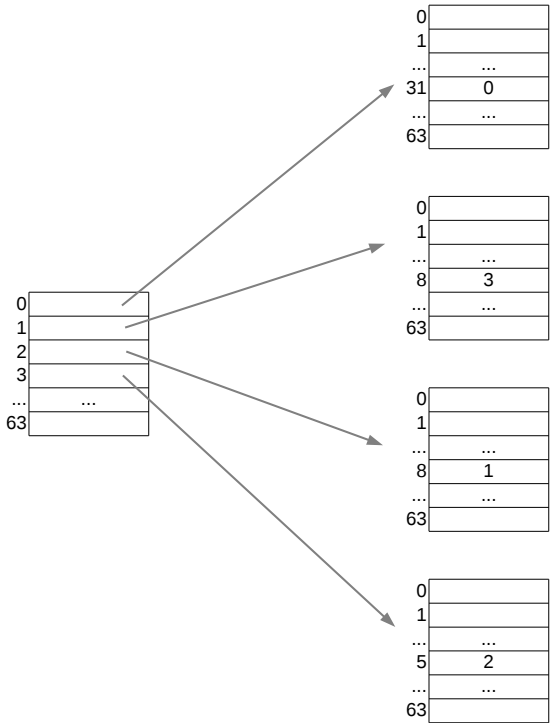

Figura 2: Tabla de páginas de dos niveles tras los accesos.