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	

UNIVERSIDAD DE MURCIA

Introducción a los Sistemas Operativos

Examen de teoría: problemas

31 de mayo de 2022

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. (1,6 puntos) Tenemos un computador con 8 GiB de RAM, memoria virtual paginada con direcciones virtuales de 56 bits, físicas de 48 bits y palabras de 64 bits. La memoria virtual usa una tabla de páginas invertida donde la función *hash* (de dispersión) para acceder a la tabla de dispersión es $h(PID, p) = (PID + p) \bmod 8$, siendo PID el identificador del proceso que hace el acceso y p la página a la que quiere acceder; las colisiones se resuelven mediante una lista ligada en la propia tabla de traducción; el algoritmo de reemplazo de páginas es NRU; en caso de empate, se reemplaza la página de menor número.

En este computador, ejecutamos un proceso con PID 1033 al que el SO ha asignado 3 marcos de memoria: 1000, 1001 y 1002. En caso de fallo de página, los marcos serán asignados inicialmente en este mismo orden. Con todos estos datos se pide:

a) (0,5 puntos) Calcular, en bytes, el tamaño de página, el tamaño de la tabla de dispersión y el tamaño de la tabla de traducción, si se sabe que esta última tiene 2^{17} entradas. Supondremos que el tamaño de cada entrada en las tablas siempre es múltiplo de palabra y que son necesarios 32 bits para guardar el PID de un proceso. Supondremos también que la entrada 0 de la tabla de traducción siempre está vacía, por lo que podemos usar dicho valor (el 0) para representar una entrada vacía en la tabla de dispersión, el final de una lista de colisiones, etc. En cuanto a posibles bits de control de las tablas, se deben incluir únicamente aquellos que sean relevantes para el ejercicio.

b) (0,8 puntos) Mostrar cómo queda la tabla de páginas invertida (tablas de dispersión y traducción) **tras cada una de las siguientes referencias a memoria virtual**, dadas también en hexadecimal: 0x0000004460D112 (escritura), 0x000001300F587A (escritura), 0x00EF00EF005543 (lectura), **tic de reloj**, 0x000001300FF001 (escritura) y 0x00000044110FF9 (lectura). Tanto la tabla de dispersión como la de traducción están inicialmente vacías. Recuerda que no se puede usar la entrada 0 de la tabla de traducción.

c) (0,3 puntos) Teniendo en cuenta la solución del apartado anterior, dibujar cómo habría quedado una tabla de páginas tradicional de dos niveles para el proceso 1033, sabiendo que el primer nivel tiene 2^{24} entradas. Recuerda trasladar a la nueva tabla **toda la información que puedas** de la tabla de páginas invertida.

---

2. (1,4 puntos) Tenemos un disco duro con 16 cabezas, 8 192 cilindros y 512 sectores por pista de 512 bytes cada uno. Usando todo el disco, se ha creado un sistema de ficheros UNIX estándar con bloques lógicos de 4 KiB. Tenemos un bloque para MBR, otro bloque para superbloque y a continuación una zona de mapa de bits de nodos-i, seguida de una zona para nodos-i (de 64 bytes cada uno) de 131 072 bloques; una zona de mapa de bits de bloques de datos seguida de la zona de bloques de datos hasta terminar el disco. El nodo-i es el estándar, con 10 entradas directas de bloque, 1 BSI, 1 BDI y 1 BTI. Usa 4 bytes para guardar un número de bloque. Con estos datos, se pide:

a) (0,6 puntos) ¿Cuál es el tamaño, en bloques lógicos, de cada una de las zonas descritas del sistema de ficheros?

b) (0,1 puntos) ¿Cuántos ficheros de cualquier tipo (regulares, directorios, enlaces, etc.) pueden crearse, como máximo, en ese sistema de ficheros?

c) (0,2 puntos) ¿Cuál sería el tamaño máximo teórico, en KiB, de un fichero con la representación de nodo-i propuesta?

d) (0,2 puntos) ¿Y el espacio máximo teórico, en KiB, ocupado por un fichero en el sistema de ficheros teniendo en cuenta sus bloques de metadatos?

e) (0,3 puntos) ¿Cuántos accesos a disco (tanto a bloques de datos como de metadatos) necesitaríamos para leer el último byte de un fichero de 4 299 202 560 bytes?

Soluciones

Ejercicio 1:

a) Como ya sabemos, una tabla de páginas invertida se compone de dos tablas: la tabla de dispersión y la tabla de traducción.

El número de entradas de la tabla de dispersión viene determinado por la función *hash*. En nuestro caso, esta función devuelve el resto de dividir el número de página virtual entre 8, por lo que, a lo sumo, podrá devolver 8 valores distintos, uno por cada entrada de la tabla de dispersión. Cada una de estas entradas almacenará el número de entrada de la tabla de traducción a la que apunte o 0 si no apunta a ninguna entrada.

En el caso de la tabla de traducción, el enunciado ya nos dice que tiene 2^{17} entradas, por lo que ha de haber ese mismo número de marcos. Como, en total, hay 8 GiB de memoria RAM, podemos concluir que el tamaño de cada página o marco ha de ser:

$$\frac{8 \cdot 2^{30} \text{ bytes}}{2^{17} \text{ marcos}} = \frac{2^{33}}{2^{17}} = 2^{16} = 65\,536 \text{ bytes/marco o página.}$$

Considerando los datos que nos da el enunciado, cada entrada de la tabla de traducción ha de tener los siguientes campos:

- El bit de validez, para saber si el contenido de la entrada es o no relevante.
- El número de la página virtual que se encuentra en el marco; para este número necesitamos $56 - 16 = 40$ bits, es decir, el tamaño de la dirección virtual menos el desplazamiento.
- El PID del proceso al que pertenece la página; según el enunciado, necesitamos 32 bits para guardar este dato.
- El número de marco donde se encuentra la página; para este número necesitamos $48 - 16 = 32$ bits, es decir, el tamaño de la dirección física menos el desplazamiento.
- Un bit de uso o referencia, para saber si la página ha sido usada o no. Este bit lo usará el algoritmo NRU.
- Un bit de modificación, para conocer si se ha modificado o no el contenido de la página y poder guardarla en disco en el caso de que sea expulsada. Este bit también lo usará el algoritmo NRU.
- El siguiente elemento de la lista en caso de colisiones. Como hay 2^{17} entradas, según el enunciado, necesitaremos 17 bits.

En teoría, no necesitamos un bit de validez, ya que, si una página no está, no se encontrará cuando se busque a través de la tabla de dispersión y de la posible lista de desbordamiento. Sin embargo, para gestionar adecuadamente la tabla de traducción (por ejemplo, para saber qué página expulsar, suponiendo que utilizamos la tabla de traducción también para esta tarea), este bit sí es necesario y por eso lo incluimos.

Cada entrada necesita, pues, $1 + 40 + 32 + 32 + 1 + 1 + 17 = 124$ bits, que son poco menos de 2 palabras. Luego cada entrada necesita 2 palabras y el tamaño total de la tabla de traducción será:

$$2^{17} \text{ entradas} \cdot 2 \text{ palabras/entrada} \cdot 8 \text{ bytes/palabra} = 2\,097\,152 \text{ bytes.}$$

En el caso de la tabla de dispersión, como cada entrada guarda la posición de la tabla de traducción a la que apunta, basta con guardar 17 bits por entrada. Sin embargo, el enunciado nos dice que el tamaño de estas debe ser múltiplo de palabra, por lo que cada entrada tendrá un tamaño de 8 bytes y el tamaño de la tabla será:

$$8 \text{ entradas} \cdot 8 \text{ bytes/entrada} = 64 \text{ bytes}.$$

Con toda esta información, la tabla de páginas invertida queda de la siguiente manera:

Estado inicial

TD

	Ptr
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	0						
2	0						
3	0						
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17}-1$	0						

Como podemos ver, en la tabla de dispersión aparece un 0 en cada entrada, indicando que no apunta a nada, y en tabla de traducción todos los bits de validez están a 0.

b) Veamos ahora cómo evoluciona la tabla de páginas invertida con cada acceso a memoria. En negrita vamos a mostrar las modificaciones realizadas en la tabla.

- Referencia: 0x0000004460D112 (escritura) → **NPV=0x4460**. El PID 1033 del proceso en hexadecimal es 409. Según la función *hash*, hay que sumar los valores NPV y PID y obtener el módulo 8 del resultado, lo que equivale a quedarse con los 3 bits menos significativos. Por lo tanto, el resultado de la función de dispersión lo podemos calcular sumando el dígito hexadecimal menos significativo del NPV y del PID, y quedándonos con los 3 bits menos significativos. Para la referencia dada, esto equivale a $h(PID, NPV) = (0 + 9) \bmod 8 = 1$. Como la tabla de dispersión y la tabla de traducción están inicialmente vacías, este acceso produce un **FALLO**. Asignamos el marco 1000 a la página, y la tabla queda así (recordemos que no podemos usar la entrada 0 de la tabla de traducción):

Tabla de páginas tras referencia: 0x0000004460D112 (escritura)

TD

	Ptr
0	0
1	**1**
2	0
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	**1**	**0x4460**	**1033**	**1000**	**1**	**1**	**0**
2	0						
3	0						
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17}-1$	0						

- Referencia: 0x000001300F587A (escritura) → **NPV=0x1300F** → $h(PID, NPV) = (F+9) \bmod 8 = 0$. La entrada 0 de la tabla de dispersión está vacía, por lo que este acceso produce un **FALLO** de página. Le asignamos a la página el marco 1001 y la tabla queda así:

Tabla de páginas tras referencia: 0x000001300F587A (escritura)

TD

	Ptr
0	**2**
1	1
2	0
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	1	0x4460	1033	1000	1	1	0
2	**1**	**0x1300F**	**1033**	**1001**	**1**	**1**	**0**
3	0						
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17}-1$	0						

- Referencia: 0x00EF00EF005543 (lectura) → **NPV=0xEF00EF00** → $h(PID, NPV) = (0+9) \bmod 8 = 1$. La entrada 1 de la tabla de dispersión apunta a la entrada 1 de la tabla de traducción, pero esta entrada no contiene información sobre la página 0XEF00EF00 y el puntero «Sig» vale 0, por lo que la lista de colisiones está vacía. Esto significa que este acceso produce otro **FALLO** de página. Le asignamos a la página el marco 1002 y la tabla queda así:

Tabla de páginas tras referencia: 0x00EF00EF005543 (lectura)

TD

	Ptr
0	2
1	1
2	0
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	1	0x4460	1033	1000	1	1	**3**
2	1	0x1300F	1033	1001	1	1	0
3	**1**	**0xEF00EF00**	**1033**	**1002**	**1**	**0**	**0**
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17}-1$	0						

- El tic de reloj pone todos los bits R a 0, dejando la tabla en el siguiente estado:

Tabla de páginas tras tic de reloj

TD

	Ptr
0	2
1	1
2	0
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	1	0x4460	1033	1000	**0**	1	3
2	1	0x1300F	1033	1001	**0**	1	0
3	1	0xEF00EF00	1033	1002	**0**	0	0
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17}-1$	0						

- Referencia: 0x000001300FF001 (escritura) → **NPV=0x1300F** → $h(PID, NPV) = (F + 9) \bmod 8 = 0$. La entrada 0 de la tabla de dispersión apunta a la entrada 2 de la tabla de traducción que contiene información sobre la página a la que se está accediendo, por lo que este acceso produce un **ACIERTO** de página. Este acierto pone a 1 los bits R y M de la página (aunque el bit M ya está a 1), dejando la tabla así:

Tabla de páginas tras referencia: 0x000001300FF001 (escritura)

TD

	Ptr
0	2
1	1
2	0
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	1	0x4460	1033	1000	0	1	3
2	1	0x1300F	1033	1001	**1**	**1**	0
3	1	0xEF00EF00	1033	1002	0	0	0
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17} - 1$	0						

- Referencia: 0x00000044110FF9 (lectura) → **NPV=0x4411** → $h(PID, NPV) = (1 + 9) \bmod 8 = 2$. La entrada 2 de la tabla de dispersión está vacía, por lo que este acceso produce un **FALLO** de página. Como todos los marcos del proceso están ocupados, hay que reemplazar una de las páginas. Si observamos cómo ha quedado la tabla de páginas en el paso anterior, la página 0xEF00EF00 pertenece a la clase 0 de NRU, ya que sus bits R y M son 0. Por lo tanto, esta es la página que se reemplaza, colocando la página 0x4411 en el marco 1002, lo que deja la tabla de páginas en el siguiente estado:

Tabla de páginas tras referencia: 0x00000044110FF9 (lectura)

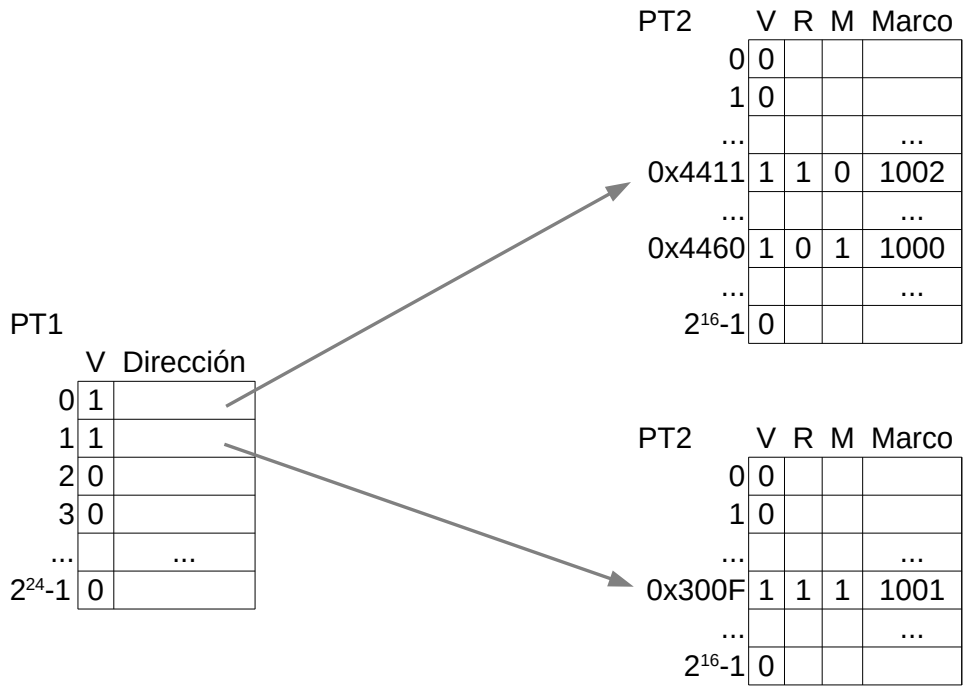
TD

	Ptr
0	2
1	1
2	**3**
3	0
4	0
5	0
6	0
7	0

Tabla de traducción

	V	NPV	PID	MF	R	M	Sig
0	0						
1	1	0x4460	1033	1000	0	1	**0**
2	1	0x1300F	1033	1001	1	1	0
3	1	**0x4411**	**1033**	**1002**	**1**	**0**	**0**
4	0						
5	0						
6	0						
7	0						
8	0						
9	0						
10	0						
...	...						
$2^{17} - 1$	0						

c) Según el apartado anterior, el marco 1000 queda ocupado por la página 0x4460, el marco 1001 por la página 0x1300F y el marco 1002 por la página 0x4411. Como la tabla de páginas es de dos niveles, los 40 bits del número de página hay que dividirlos en dos campos. El enunciado nos dice que el primer nivel tiene 2^{24} entradas, por lo que el primer campo tiene un tamaño de 24 bits y el segundo tiene los 16 bits restantes. O lo que es lo mismo, los 6 dígitos hexadecimales más significativos de un NPV nos dan la entrada de la primera tabla a usar y los 4 dígitos hexadecimales menos significativos nos dan la entrada de la segunda tabla (cada tabla de segundo nivel tiene 2^{16} entradas). Con estos datos, la tabla de segundo nivel queda como refleja la siguiente figura:



Ejercicio 2:

A continuación se muestran las soluciones de los diferentes apartados.

a) En primer lugar hay que determinar el tamaño del disco, para lo cual multiplicamos las cabezas, por los cilindros, por los sectores que tiene cada cilindro y por 512 bytes por sector, lo dividimos entre 1 024 y nos da los KiB del disco: 33 554 432 KiB, que equivalen a 32 GiB. Dividiendo entre 4, obtenemos los bloques lógicos del disco: 8 388 608 bloques. El primer bloque es el MBR, el segundo el superbloque, tenemos dedicados 131 072 bloques para nodos-i, como cada uno tiene 64 bytes, podemos calcular el número de nodos-i que hay: $(131\,072 \times 4\,096)/64 = 8\,388\,608$ nodos-i. Esto quiere decir que cada uno de esos nodos-i tiene un bit en su mapa de bits, y por tanto necesitaremos: $8\,388\,608/(8 \times 4\,096) = 8\,388\,608/32\,768 = 256$ bloques. Esto nos lleva a que la zona de mapa de bits de bloques de datos más la zona de bloques de datos tienen que ocupar $8\,388\,608 - 1 - 1 - 256 - 131\,072 = 8\,257\,278$ bloques. Si llamamos x al número de bloques lógicos ocupados por el mapa de bits de bloques de datos, podemos plantear la siguiente ecuación:

$$x + 32\,768 \cdot x = 8\,257\,278$$

Despejando la x, obtenemos:

$$x = \frac{8\,257\,278}{32\,769} = 251{,}98 = 252$$

Por tanto el mapa de bits de bloques de datos ocupará 252 bloques y la zona de bloques de datos tendrá un tamaño de $8\,257\,278 - 252 = 8\,257\,026$ bloques.

b) Se podrán crear tantos como nodos-i tengamos, por tanto serán 8 388 608 ficheros o directorios.

c) Si los bloques lógicos son de 4 KiB y una dirección de disco ocupa 4 bytes, cada bloque indirecto podrá almacenar 4096/4=1024 direcciones. Con este dato, el tamaño máximo teórico de un fichero con esta implementación de nodos-i sería aquel que hiciera que el fichero ocupara todo su árbol de metadatos, teniendo por tanto: $10 + 1\,024 + 1\,024^2 + 1\,024^3$ bloques. Con bloques de 4 KiB, el tamaño máximo teórico sería 4 299 165 736 KiB, es decir, algo más de 4 TB.

d) El tamaño de los metadatos (sin contar el nodo-i), sería el que ocupen los BSI, BDI y BTI del fichero anterior. Como tenemos 1 BSI, 1 BDI y 1 BTI completamente llenos, eso nos da un total de: $1+1+1\,024+1+1\,024+1\,024^2 = 1\,050\,627$ bloques de metadatos, que si los sumamos al número de bloques de datos, dan un total de: 4 303 368 244 bloques.

e) Si el fichero tiene un tamaño de 4 299 202 560 bytes, su último byte será el 4 299 202 559. Si dividimos este número entre 4 096, tendremos el número de bloque que hay que leer, es decir, el $1\,049\,609{,}99$, o lo que es lo mismo, el $1\,049\,609$. Ahora podemos restar los 10 bloques del nodo-i y calcular en qué BSI se encuentra el número de bloque de disco correspondiente a ese bloque del fichero: $(1\,049\,609-10)/1\,024 = 1\,024{,}999 = 1\,024$ empezando a contar desde 0. Como el BSI 0 es el que está independiente, tendremos que llegar al BSI 1 023 a partir del BDI. Como el BDI contiene 1 024 BSIs, allí estarán desde el 0 al 1 023 y, por tanto, estaremos accediendo al último elemento del BDI. Esto nos da un total de 1 acceso para leer el BDI cuya dirección está en el nodo-i, 1 acceso para leer el BSI (cuya dirección se encuentra en ese BDI) y un acceso para leer el bloque de datos, es decir, 3 accesos si no contamos el acceso a nodo-i para comenzar.