

SOA2 (19/05/2025)

Nombre:

DNI:

Segundo control de teoría

Responde brevemente a las siguientes preguntas justificando tus respuestas. Una respuesta sin justificar no será dada como válida. Todas las preguntas tienen la misma puntuación.

En el cierre de ayer de la bolsa española, la empresa Baka Corp., administrada por el Señor Baka Baka, casi perdió un 10% de su cotización. Según los expertos, esta pérdida se debe a dos motivos: el primero es la nueva condena a 10 años de prisión del Señor Baka Baka por un delito recurrente de maltrato psicológico a los estudiantes de SOA y de SO2. El segundo motivo es porque su nueva versión del sistema de ficheros de ZeOS, llamado ZeFS, presenta un rendimiento mucho más bajo del esperado.

ZeFS tiene las siguientes características:

- La estructura de directorios es en grafo
- Utiliza una asignación de sectores de disco a bloques de datos con bloques iguales que no comparten sectores
- Todos los bloques de datos son de 16 sectores
- Asignación indexada de bloques a ficheros
- El Inodo tiene el mismo tamaño que el bloque de datos
- Tiene un MBR en el primer sector del disco
- El superblock está en el primer bloque de la partición y solo ocupa un bloque

En ZeFS los identificadores de Inodos y de bloques de datos ocupan 4 bytes. ZeFS solamente se puede ejecutar en máquinas de 32 bits, por tanto, el tamaño del tipo entero escalar más grande es 4 bytes y siempre se utilizará este tipo cuando se quiera guardar un valor numérico.

El disco duro en el que se están haciendo las pruebas tiene una capacidad de 500 GBytes y un tamaño de sector de 256 bytes. En el primer sector del disco se aloja el MBR.

Pregunta 1

¿Es necesario implementar Inodos en ZeFS?

Sí porque el SF tiene estructura de grafo ✓

Pregunta 2

¿Qué información tiene que guardar el superblock en ZeFS?

Contiene información sobre qué bloques de datos y inodos están libres o ocupados X

* Sectores por bloque * Num de inodos * Bitmap de inodos libres
* Num de bloques por partición en la partición * Bitmap de bloques libres

De entre estos campos, hay unos cuyo tamaño dentro del superblock dependen del tamaño de la partición.

Pregunta 3

¿Qué campos del superblock varían su tamaño dependiendo del tamaño de la partición?

Bitmaps de inodos y de bloques de datos libres ✓

Para hacer las siguientes preguntas independiente de las anteriores, fijaremos, a partir de ahora, que el espacio dentro del superblock ocupado por los campos de tamaño fijo es de 40 bytes.

Pregunta 4

¿Calcula cuál será el tamaño máximo de una partición en ZeFS con el disco duro de pruebas?

$$(500\text{GB} - 256\text{bytes}) / 4 = 124.999.936\text{bytes} \approx 125\text{ GB} \times$$

* 1 bloque tiene 16 sectores $\rightarrow 16 * 256 = 4096\text{bytes} / \text{bloque}$ * $32448\text{block/part} * 16\text{sect/block} * 256\text{bytes/sect.} =$
* $4096 - 40 = 4056\text{bytes para bitmaps}$ = $132.907.008\text{bytes/partición + bloques superblock}$
* $4056 * 8 = 32448\text{bits (bloques totales por partición)}$ ↴ superblock

Pregunta 5

¿Cuántas particiones de tamaño máximo se pueden tener como máximo en el disco de pruebas?

4 particiones, las que permite el MBR ✗

Realmente, la respuesta anterior demuestra que hemos encontrado el primer problema de ZeFS: no se puede utilizar todo el espacio de almacenamiento de un disco actual. Al comunicárselo al Señor Baka Baka, se pone a llorar desconsolado pensando en la cantidad de dinero que ha invertido en este sistema de ficheros y acto seguido empieza a escribir una lista de los ingenieros a los cuales va a despedir. Pero, de repente, se calma, deja de llorar y te pregunta...

Pregunta 6

¿Qué modificación sencilla se podría hacer para utilizar todo el espacio de almacenamiento de un disco duro actual?

- * Aumentar el tamaño del superblock para que sea más de un bloque de datos
- * Aumentar el nro de sectores por bloque de datos

Para que las siguientes preguntas sean independientes a las anteriores, a partir de ahora fijaremos a 64Mbytes el tamaño máximo de una partición en ZeFS, sin contar el superblock.

Pregunta 7

Suponiendo que hay tantos Inodos como bloques de datos, ¿cuántos Inodos tendrá como máximo una partición de tamaño máximo en ZeFS?

$$64\text{Mbytes} * 8\text{bytes/byte} = 512.10^6\text{inodos} \times 1\text{block} * 16\text{sect/block} * 256\text{bytes/sector} = 4\text{KB/block}$$

* $64\text{MB} / 4\text{KB} = 16.384\text{bloques/partición} \rightarrow 50\% \text{inodos}, 50\% \text{datos} \rightarrow 8.192\text{bloques inodo}$

El porcentaje de fragmentación interna dentro de un bloque de datos se calcula con la fórmula:

$$\% \text{Fragmentacion interna} = (\text{Bytes no usados} * 100) / \text{Bytes totales}$$

Pregunta 8

Si la partición es de 10Mbytes, sin contar el bloque de datos del superblock, ¿Cuál es el porcentaje de fragmentación interna dentro del bloque de datos que contiene el superblock?

$$10\text{MB} / 4\text{KB} = 2560\text{bloques/partición indexados (bit inodo)} \quad 1\text{bloque} \rightarrow 4096\text{bytes totales}$$

~~$1\text{bloque} * 16\text{sect} * 256\text{bytes} = 4.096\text{bytes totales}$~~ #bytes en superblock = $40\text{(cabecera)} + 2560\text{bytes} / (8\text{bytes/byte}) = 360\text{bytes}$
 $\% \text{frag. interna} = ((4096 - 360) * 100) / 4096 = 91,21\%$

SOA2 (19/05/2025)

Nombre:

DNI:

Uffff... demasiado espacio perdido dentro del superblock para una partición de 10Mbytes... pero esta vez no se lo diremos al Señor Baka Baka porque corren rumores que está deambulando por las oficinas de la empresa con una pistola en la mano.

Nosotros a lo nuestro. Para que las siguientes preguntas sean independientes a las anteriores, vamos a suponer que la partición de tamaño máximo tiene 8192 Inodos y 8192 bloques de datos.

Pregunta 9

Si cambiamos la asignación indexada por asignación contigua y suponiendo que la partición en la cual estamos haciendo pruebas, está vacía, y es de tamaño máximo, ¿Cuál es la máxima cantidad de datos del usuario, en bytes, que puede tener un fichero?

1 fichero de tamaño máximo. $8192 \text{ bloques de datos} = 8192 * 16 * 256 = 33.554.432 \text{ bytes}$ ✓

Pregunta 10

¿Cambia la respuesta anterior si se cambia la asignación contigua por una encadenada (sin tabla)?

Si porque cada bloque de datos tendría 4bytes menos ya que se usarían para indexar el próximo bloque.

$\rightarrow 8192 \text{ bloques} * (4096 - 4) \text{ bytes/bloq} = 33.521.664 \text{ bytes}$ ✓

Pregunta 11

¿Cuál es la máxima cantidad de datos del usuario, en bytes, que puede tener un fichero con la asignación indexada original?

Igual que en la contigua, ya que los bloques de datos se usan igual x Tenemos bloques de índices entre los de datos

Cada bloque de índices guarda $4096 \text{ bytes} / (4 \text{ bytes/bloque aniquila}) = 1024 \text{ bloques} \rightarrow 8192 / 1024 = 8 \text{ bloques de índices}$

$\# \text{bytes de datos} = (8192 - 8) * 4096 \text{ bytes/bloq} = 33.488.928 \text{ bytes}$

Pregunta 12

Indica cuantos accesos a disco, en caso de asignación contigua, se tiene que hacer para leer el byte 143460 en caso de tener dentro de la partición un único fichero de tamaño máximo.

1 único acceso ya que hay correspondencia directa entre n°byte y n°bloque ✓

Pregunta 13

Indica cuantos accesos a disco, en caso de asignación encadenada, se tiene que hacer para leer el byte 143460 en caso de tener dentro de la partición un único fichero de tamaño máximo.

Hemos de recorrer todos los anteriores bloques de forma secuencial, así que habrá que hacer

$143.460 / 4096 = 35.02 \approx 35 \text{ accesos a disco}$

Pregunta 14

Indica cuantos accesos a disco, en caso de asignación indexada, se tiene que hacer para leer el byte 143460 en caso de tener dentro de la partición un único fichero de tamaño máximo.

Tendremos que acceder al bloque de índice correspondiente y recorrerlo secuencialmente.

+ 1 acceso al bloque de datos ≈

Mientras estabas calculando el tamaño máximo de un fichero con la asignación es indexada, se te ha ocurrido una idea para reducir el número de bloques de índices que tiene que utilizar un fichero. Corriendo, vas a contársela al ingeniero jefe de ZeFS pero éste ha desaparecido misteriosamente. Así que decides implementar y probar la siguiente optimización: en los bloques de índices, en vez de guardar los índices a bloques de datos, se guardarán parejas <índice de bloques de datos inicial, número de bloques consecutivos>, de tal forma que se podrán asignar regiones consecutivas libres del disco ocupando menos espacio dentro del bloque de índices.

Pregunta 15

Suponiendo que la partición de pruebas está vacía y es de tamaño máximo, ¿Cuál es el tamaño máximo de fichero con esta optimización?

Solo habrá un índice (1 índice fichero), por tanto tendremos 8 bloques de índices solamente.

$$\text{Tamaño máx. fichero} = (8192 - 1) * 4096 = 33\ 550\ 336 \text{ bytes. } \checkmark$$

Pregunta 16

Plantea un escenario en el que la optimización planteada necesite más bloques de índices para almacenar un fichero que la asignación indexada normal.

Con fragmentación externa. Por ejemplo los bloques de datos para están todos asignados y queremos crear un fichero igual de grande que el espacio libre. ✓

Pregunta 17

¿Cómo se podría solucionar el escenario que has descrito?

Con un proceso de defragmentación

Pregunta 18

El acceso aleatorio con esta optimización, ¿es mejor que el de la asignación indexada?

No siempre, además genera más fragmentación en mucho peor ✓

ZeFS, como cualquier otro sistema de ficheros en ZeOS, es abstraído por un Virtual File System. Este VFS implementa un gestor multiprogramado (con varios threads que procesan peticiones). Por ahora supón, a no ser que se diga lo contrario, que la entrada/salida siempre es síncrona.

Pregunta 19

¿Qué estructuras del sistema de ficheros son accedidas para enviar una petición a este gestor y recibir el resultado?

* Tabla de carpetas

* Descriptor de disp. de gestor

* Tabla de ficheros abiertos

* Cola iorbs (input/output request block)

✗

* Tabla de nodos

* Cola io-fin

SOA2 (19/05/2025)

Nombre:

DNI:

Pregunta 20

¿Cuántos semáforos se tienen que implementar para poder enviar una petición al gestor?

4 : Uno por cola (2 colas), 1 por petición y 1 por gestor ✓

Pregunta 21

¿Qué tamaño deben tener las colas para enviar peticiones al gestor, con la configuración actual de ZeOS (la que habéis visto en el laboratorio) para asegurarnos de que no se gasta memoria innecesariamente?

Como tenemos 10 procesos/thread máximos. 10 iobks + 10 io_fin

Pregunta 22

¿Cambia la respuesta anterior en el caso de que también se permita hacer entrada/salida asíncrona?

No porque va ligado a wantos TCBs tenemos X Si porque en el mismo thread puede tener más de una petición pendiente

Pregunta 23

Escribe el código del gestor de VFS

```
while (1) {  
    sem_wait(sem_gestor);  
    coser_peticion();  
    Do I/O;  
    encolar_resultado();  
    sem_signal(sem_peticion);  
}
```

Pregunta 24

Escribe el código de la función read dependiente de VFS en caso de tener entrada/salida asíncrona.

```
int vfs_read() {  
    encolar_peticion();  
    sem_wait(sem_gestor)  
    sem_wait(sem_gestor)  
    sem_post(sem_gestor);  
}
```

Pregunta 25

¿Se tiene que añadir algo más para implementar entrada/salida asíncrona?

La llamo a sistema isReady para saber cuando ha acabado el gestor

Pregunta 26

Te tienes que asegurar que, desde el punto de vista de optimizaciones, se han aplicado todas las que hemos visto en clase de teoría. ¿Qué estructuras de datos dentro del código del sistema de ficheros tienes que buscar para asegurarte de este hecho?

Tabla de inodos y buffer caché

Como has visto anteriormente, ZFS solamente funciona con discos duros antiguos. Debido a su tecnología hardware, estos discos duros no permiten realizar más de un acceso simultáneo para leer y/o escribir datos. Cada vez que se accede al disco, solamente se puede leer o escribir un sector. Cada disco duro tiene su propio gestor.

Pregunta 27

¿A qué estructura de datos se tiene que acceder para poder ejecutar la función hdu_read que es donde se implementa la lectura de una sección de disco duro?

A la buffer caché, probablemente lo tengo cargado X Al descriptor de dispositivo

Si la llamada al sistema:

```
int sys_read(int fd, void *buffer, int size);
```

llama al read dependiente del VFS:

```
int vfs_read(Inode_t inode, void *buffer, int size);
```

y ésta acaba llamando, antes o después a hdu_read para acceder al disco...

Pregunta 28

Indica la cabecera, justificando cada uno de sus parámetros, de la función hdu_read

int hdu_read ((void*)dev_dap, void *buffer, int size);
↑
Puntero al dispositivo de disco
Dev_t device
↓
Disco duro del que se tiene que leer
↑
Buffer con contenido de disco (mismo tamaño que un sector)
↑
Tamaño del texto leído
↓
Sector del disco duro

Mirando el código del gestor de disco duro, te das cuenta de que es multiprogramado y, echándote las manos a la cabeza gritas: "NO!". Pero el caos reina en la oficina en la que estás trabajando y se ha producido un incendio entre las plantas 15 y 20 del edificio de Baka Corp que hace que tu integridad física peligre. Aun así, decides seguir trabajando.

Pregunta 29

¿Por qué no es correcto que el gestor de disco duro sea multiprogramado?

Porque entonces distintos programas podrían acceder simultáneamente al mismo bloque creando condiciones de carrera

Porque el disco solo admite 1 petición simultánea (enunciado)

X

SOA2 (19/05/2025)

Nombre:

DNI:

Viendo que el gestor de disco tiene que ser monoprogramado...

Pregunta 30

¿Sería recomendable cambiar también el gestor de VFS para que sea monoprogramado?

No porque se pueden acceder a distintos dispositivos simultáneamente ✓

Y a todo esto...

Pregunta 31

¿Cuál es la función del gestor de ZeFS?

Manejar las operaciones de E/S del sistema garantizando integridad en los bloques de datos del disco X

Entregado de señores la partición traduciendo los accesos genericos al VFS a accesos de disco que enviarán al buffer cache (si existe)

Ya tienes toda la información que necesitas para acabar de redactar tu informe sobre ZeFS. Exhausto, y tosiendo salvajemente debido al humo, subes al despacho del Señor Baka Baka el cual bebe de una botella de whiskey. Le gritas: "¡¡¡Señor Baka Baka, no está todo perdido!!! ¡¡Hay que hacer modificaciones en ZeFS pero tampoco son tantas!!! ¡Podemos remontar esta situación!". El Señor Baka Baka se gira hacia ti y con unos ojos llenos de optimismo, te pregunta...

Pregunta 32

¿Qué modificaciones se tienen que hacer?

- * El superblock ha de ser de más de un bloque
- * Hacer que la E/S sea asíncrona
- * Discos duros modernos que permitan acceso concurrente
- * Implementar asignación indexada por páginas con defragmentación incorporada

Información del Sistema Operativo

El sistema dispone de las siguientes rutinas:

- `struct task_struct *current()`: Retorna la task_struct del proceso actual.
- `int alloc_frame()`: Reserva un frame de memoria física.
- `void free_frame (int frame)`: Libera un frame de memoria.
- `page_table_entry * get_PT(struct task_struct *t)`: Retorna la tabla de páginas del proceso *t*.
- `page_table_entry * get_DIR(struct task_struct *t)`: Retorna el directorio de páginas del proceso *t*.
- `set_CR3 (page_table_entry * dir)`: Sobreescribe el registro CR3 con el nuevo directorio de páginas *dir*, provoca una invalidación de la TLB.
- `int get_frame (page_table_entry *pt, int logical_page)`: Retorna el frame asignado a una página lógica en la tabla de páginas de un proceso determinado.
- `int set_ss_page (page_table_entry *pt, int logical_page, int frame)`: Asigna un *frame* a una página lógica de la tabla de páginas *pt*.
- `int del_ss_page (page_table_entry *pt, int logical_page)`: Borra una entrada de la tabla de páginas.