

## Memorias Flash y el Sistema de Ficheros LogFS

**COMO UN RAYO**

Actualmente la tecnología Flash forma parte de los entornos Linux. El nuevo sistema de ficheros LogFS ayudará al usuario frente a los problemas típicos de las memorias flash.

**POR JAN KLEINERT Y ACHIM LEITNER**

Las memorias flash estaban reservadas para las aplicaciones “incrustadas”. Con el paso del tiempo, sin embargo, los precios han bajado drásticamente para flash en forma de memorias USB, así como en forma de tarjetas de memoria para las cámaras, las PDAs, los teléfonos móviles, los discos y los reproductores de MP3. Los usuarios constantemente tienen la necesidad de acceder a los dispositivos basados en memorias flash desde sus sistemas Linux, de ahí que la comunidad Linux esté respondiendo con mejores herramientas para abordar los retos que presentan estas memorias.

LogFS ha entrado recientemente en la carrera de los sistemas de ficheros Linux para las memorias Flash.

### La Capa de Traducción Flash

En los sistemas actuales, una aplicación normal puede que incluso no sea consciente de si está o no utilizando una memoria flash o un sistema de ficheros ordinario. La responsabilidad de la comunicación con el dispositivo está firmemente fijada en el hardware o en el kernel. Alrededor del

año 2000 se presentó una Capa de Traducción Flash (FTL) en el kernel. La capa gestiona los requerimientos especiales a la hora de direccionar la memoria flash.

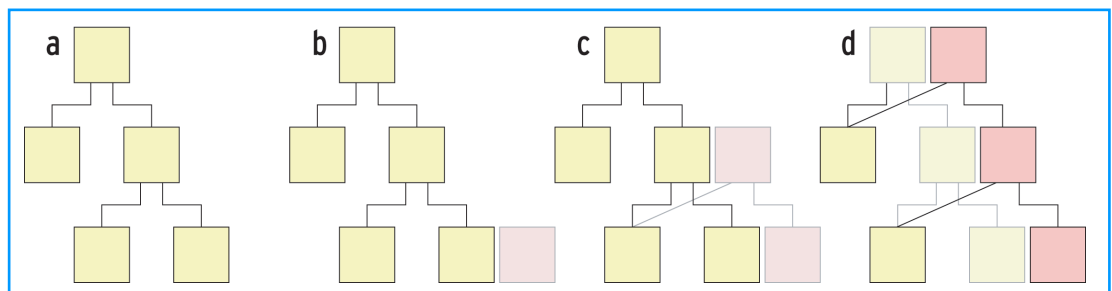
FTL está diseñada para permitir a los sistemas de ficheros leer y escribir los sectores sin conocer cómo está organizada la memoria y sin tener que molestarse en borrar antes de escribir (Véase el cuadro “Tecnología Flash”). En la actualidad, el kernel de Linux soporta cinco variantes FTL en el subsistema de memoria.

Como los bloques de los sistemas de ficheros son mucho más pequeños que los bloques de la memoria flash, FTL optimiza los accesos de escritura acumulando los cambios para ahorrar los ciclos de borrado y escritura, mejorando de este modo la longevidad y el rendimiento del chip. Esta técnica

requiere estructuras de gestión (como la reasignación de bloques) que colaboren con el recolector de basura. El controlador se encarga de manejarlo en el caso, por ejemplo, de una tarjeta Compact Flash (CF) o de una memoria USB, sirviendo una interfaz ATA normal en el sistema. Los sistemas de ficheros ext2 o FAT no están optimizados para usarse con las memorias flash. La FTL ni siquiera conoce si un bloque ha sido borrado, ya que el sistema de ficheros no pasa esta información a la capa del bloque. Esto hace que la recolección de basura sea aún más difícil.

### Presentación de LogFS

Los sistemas de ficheros especiales como JFFS2 (Journaling Flash File System 2 [1], un desarrollo basado en el original JFFS [2]),



**Figura 1: LogFS escribe los cambios fuera de lugar. En la estructura de árbol (a) se crea un bloque modificado en una posición nueva (b) y deja el bloque antiguo sin modificar. El cambio aún no se enlaza y por ello LogFS tiene que crear los nodos padres (c). La estructura nueva no se aplica hasta que el nodo raíz se haya procesado (d). De este modo se consigue un estado consistente en el sistema de ficheros.**

UBIFS [3] o YAFFS [4] dan un paso hacia adelante a la hora de enfrentarse a las complicaciones que presentan las memorias flash. El nombre "Journaling Flash Filesystem" es desafortunado, ya que JFFS realmente utiliza una estructura basada en registros. Una nueva estrella en esta parte del cielo de Linux es LogFS [5], que no debe confundirse con el proyecto ya abandonado LFS [6] (Log Structured File System).

### LogFS vs JFFS2

Jörn Engel, el encargado de mantener LogFS, está posicionando a LogFS contra JFFS2. Engel espera ganar puntos frente a JFFS2 en campos como el rendimiento. Por ejemplo, dice que se tardan cerca de 15 minutos en montar una memoria USB de 1GB con JFFS.

### Moviendo Árboles

La idea central tras LogFS es su estructura para la gestión de árboles. LogFS no borra y escribe los bloques modificados, aunque coloca el contenido al final del área de la memoria utilizada. Por ello, acumula los cambios que cubren segmentos de varios bloques borrados, como los métodos de recolección de basura de las FTD.

Para actualizar la estructura del árbol, LogFS reescribe todas las ramas del bloque modificado hasta la raíz al final de la memo-

ria flash en uso, de esta manera declara implícitamente todos los bloques previos utilizados como memoria libre (Figura 1). Al montar un dispositivo LogFS, el controlador busca el último nodo raíz. La estructura de datos será consistente bajo este nodo. Esta solución ahorra operaciones de borrado y de escritura.

### Uso de LogFS

Para utilizar LogFS hay que parchear el código fuente del kernel oficial. El sitio web de LogFS [5] contiene parches para Linux 2.6.18, 2.6.20, 2.6.21 y 2.6.23. La herramienta `mklogfs` está disponible desde la misma fuente.

Tras la configuración, compilación y arranque se puede crear un MTD [7] (Memory Technology Device) `mklogfs /dev/mtd0` y montar el dispositivo con el sistema de ficheros: `mount mtd0 /mnt -t logfs`.

LogFS se encuentra actualmente en desarrollo. Jörn Engel dice que su sistema de ficheros aún actualiza el árbol con demasiada frecuencia [5], y por ello el rendimiento se ve afectado. Sólo podemos desearle buena suerte con LogFS.

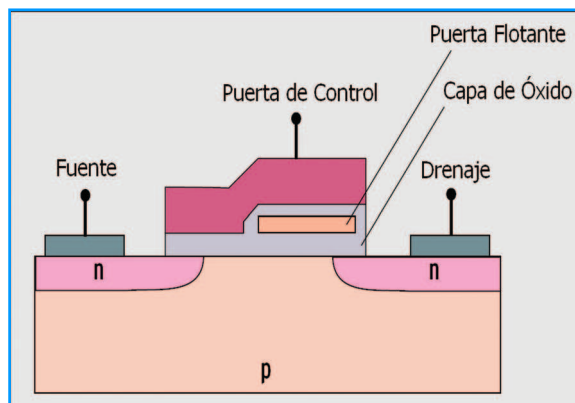


Figura 2: La capa de óxido que rodea la puerta flotante impide que los electrones se escapen. (Fuente de la figura: Wikipedia).

### Soluciones Alternativas

Los desarrolladores utilizan diversas soluciones para abordar las complicaciones que presenta la arquitectura flash. Este artículo ha mostrado cómo tratar los problemas de las técnicas flash en la capa de bloques (con la FTL) y con el sistema de ficheros (con LogFS).

Otras opciones tienen que tratar estos problemas a nivel del dispositivo o diseñar el soporte directamente en la aplicación, una técnica usada a menudo con aplicaciones empaquetadas altamente especializadas.

## Tecnología Flash

Las EEPROMs Flash son chips de memorias con contenido no volátil, es decir, retienen los datos aunque no estén conectados a una fuente de alimentación. Los chips flash NAND que conectan cierta cantidad de transistores flash (puertas flotantes) en serie son usados habitualmente por su pequeño tamaño y su bajo coste de fabricación. Como las operaciones de escritura selectivas sólo conmutan el estado de falso a verdadero, se requiere una operación lógica anterior a cada escritura para introducir un ciclo de borrado.

### Borrado de Bloques

Al contrario que las memorias EEPROM normales, en una EEPROM Flash no se puede borrar una simple palabra, unidad de memoria (de 8 a 64 bits) individual direccionable más pequeña, sino que hay que borrar un bloque. Un bloque normalmente es un cuarto, un octavo, una dieciseisava, ... parte de la capacidad de memoria total del chip. Estos bloques EEPROM (también conocidos como bloques de borrado) son mucho mayores que los bloques que habitualmente utilizan los sistemas de ficheros. Al mismo tiempo, el número de ciclos de borrado (resistencia) es limitado. Los fabricantes no garantizan más de 100.000 a un millón de ciclos para cada bloque.

La razón es que los electrones de las puertas flotantes atraviesan la capa de óxido del transistor en el borrado (un fenómeno conocido como efecto túnel de Fowler-Nordheim). El alto voltaje requerido para este efecto mecánico cuántico daña ligeramente la capa de óxido que rodea la puerta flotante cada vez que se realiza un borrado (véase la Figura 2). Una vez que el proceso de degeneración ha llegado a cierto umbral, los electrones atrapados en el transistor empiezan a escapar; el bit almacenado en esta posición se pierde, y la celda de memoria se queda defectuosa. Los fabricantes de dispositivos flash intentan contrarrestar este efecto incluyendo celdas de reserva y reasignando los bloques defectuosos.

## Flash NAND

### Ventajas:

- Tiempos de acceso cortos
- Bajo consumo de energía
- Retención de los datos sin corriente
- Resistente a los golpes y a los campos magnéticos

### Desventajas:

- Operaciones de escritura lentas
- Sólo borrado de sectores
- Número limitado de ciclos de escritura
- Más caras que las alternativas

## RECURSOS

- [1] JFFS 2: <http://sources.redhat.com/jffs2/>
- [2] JFFS: <http://developer.axis.com/old/software/jffs/index.html>
- [3] UBIFS: <http://www.linux-mtd.infradead.org/doc/ubifs.html>
- [4] YAFFS: <http://www.yaffs.net>
- [5] LogFS: <http://www.logfs.org/logfs/>
- [6] LFS: <http://logfs.sourceforge.net>
- [7] MTD: <http://linux-mtd.infradead.org>