Block fileSystems

Mg. Ing. Gonzalo E. Sanchez
MSE - 2020

Implementación de Sistemas Operativos II

Block fileSystems

- Introducción

Introducción

Block fileSystems

- Los dispositivos de almacenamiento se clasifican en dos grandes grupos:
  - Dispositivos block.
  - Dispositivos flash.
- Son manejados por subsistemas diferentes y por filesystems diferentes.
- Los dispositivos block pueden ser escritos y leídos en operatorias de bloque, sin ser borrados previamente.

Block fileSystems

- Ejemplos de dispositivos block:
  - Discos duros.
  - Disquettes.
  - Discos RAM.
  - Pen Drives.
  - Memorias SD
- Pendrives y memorias SD se basan en sistemas flash pero integran un controlador que emula un dispositivo block.

Block fileSystems

- Los dispositivos flash pueden ser leídos pero para su escritura requieren ser borrados previamente.
- Normalmente se debe borrar una porción más grande del tamaño que se desea escribir.
- Ejemplos de dispositivos flash:
  - Memorias NOR flash.
  - Memorias NAND flash.

Block fileSystems

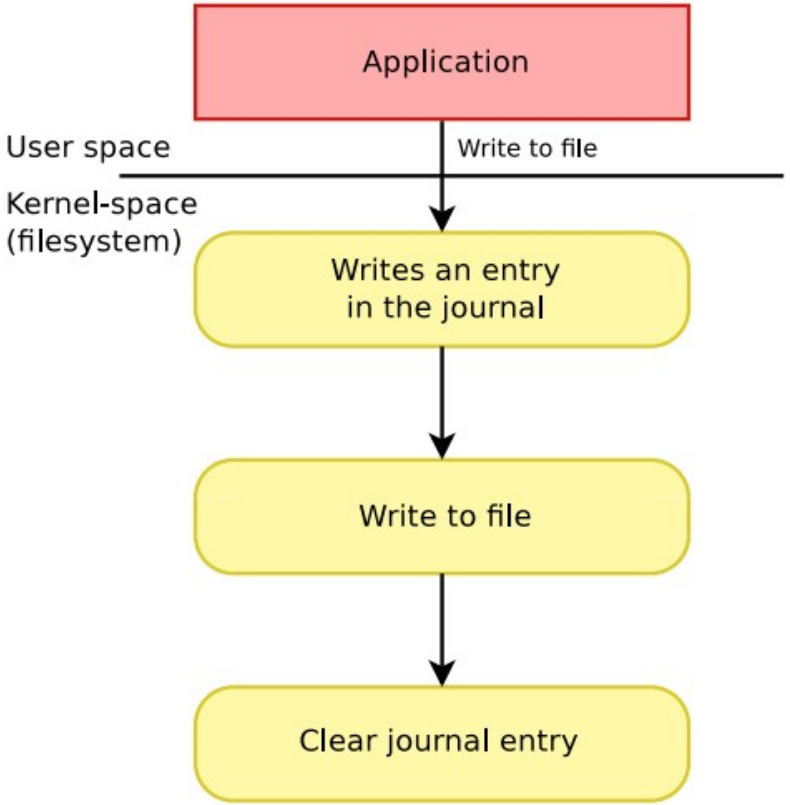
- La lista de dispositivos block disponibles en el sistema se pueden encontrar en **/proc/partitions**.
- También puede verse en **/sys/block**.
- Usualmente los discos duros (y sus respectivas particiones) se muestran como **sda**, **sdb**, etc.
- También puede verse los pendrives representados como **sda**, **sdb**, **sdc**, etc.
- Memorias SD suelen mostrarse como **mmcblk** (si no usan un adaptador)

Block fileSystems

- Los filesystem tradicionales luego de un system crash o desconexión brusca pueden quedar en un estado no coherente.
- Esto requiere un chequeo completo del filesystem luego del reinicio.
- Sistema de archivos tradicional para linux: **ext2.**
- Sistema tradicional para Windows: **vfat**.
- Herramientas de reparación: **fsck.ext2** y **fsck.vfat** respectivamente.

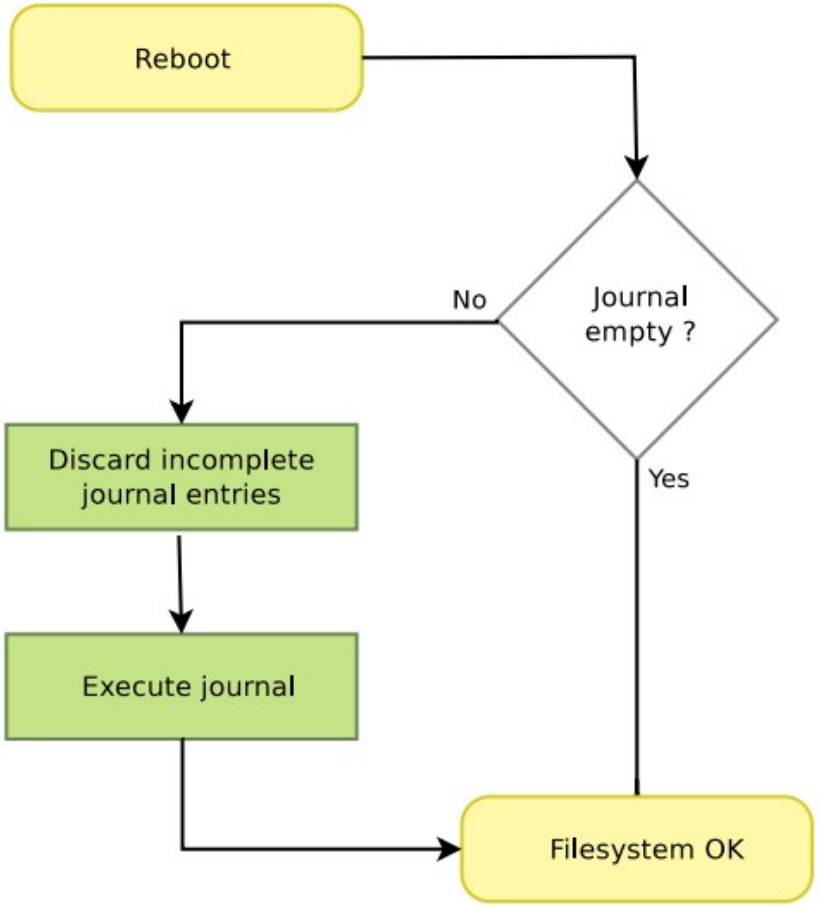
Block fileSystems

- Para evitar estados no coherentes en condiciones de poweroff brusco o luego de un system crash, se diseñaron otros **fs**.
- Son denominados *journaled filesystems*.
- Todas las escrituras en ellos son descritas antes en un diario o bitácora.



Block fileSystems

- Gracias a la bitácora, el filesystem jamás se deja en un estado corrupto/inválido.
- Con un poweroff repentino puede llegar a perderse información recientemente guardada.



Block fileSystems

- Filesystems que incluyen bitácora:
  - **ext3**: es un filesystem ext2 con extensión de bitácora.
  - **ext4**: nueva generación con muchas mejoras. Es el filesystem por defecto de todos los sistemas linux actuales.
- El kernel de linux soporta muchos otros filesystems:
  - reiserFS.
  - JFS.
  - XFS.
- Orientados a servidores o cargas de trabajo científicas.

Block fileSystems

- Existe un posible reemplazo para ext4 en el futuro cercano: btrfs.
- Filesystem de próxima generación. Excelente performance
- Todavía bajo fuerte desarrollo (link a wiki oficial).
- El formato en disco es estable: no se prevé un cambio en él salvo existan razones muy fuertes para hacerlo.
- Conferencia sobre btrfs explicando pros y contras (link).

Block fileSystems

- Para particiones muy pequeñas (<5MB) se recomienda formato ext2.
- Esto es porque ext3 y ext4 necesitan mucho espacio en proporción para metadata (1MB para una partición de 4MB).
- Para crear una particion vacia ext2/ext3/ext4 en un dispositivo block se utiliza el comando mkfs:
  - **mkfs.ext2 /dev/hda3**
  - **mkfs.ext3 /dev/sda2**

Block fileSystems

- Para crear una imagen de filesystem a partir de un directorio que contiene archivos y directorios se utiliza **genext2fs**.
- **genext2fs -d rootfs/ rootfs.img**
- Así podremos transferir la imagen al dispositivo block mediante el comando **dd**.
- También puede accederse y modificarse utilizando el mecanismo **loop**.

Block fileSystems

- Ejemplo:
  genext2fs -d rootfs/ rootfs.img
  mkdir /tmp/tst
  mount -t ext2 -o loop rootfs.img /tmp/tst
- En el directorio **/tmp/tst** se puede acceder y modificar los contenidos de **rootfs.img**
- Posible gracias a un driver del kernel llamado **loop** que emula un dispositivo block a partir de los contenidos de un archivo.

Block fileSystems

- Existen filesystems optimizados para ser utilizados en block devices basados en NAND flash.
- Flash-Friendly filesystem es uno (link a wiki F2FS).
- Disponible en el mainline de linux.
- Muy buenos benchmarks, la mejor performance en la mayoria de los casos.

Block fileSystems

- Un tipo de filesystem muy conocido es **squashfs**.
- Es un filesystem de solo lectura para dispositivos block.
- Es adecuado para partes de un filesystem que solo pueden ser leídos (kernel, binarios de sistema, etc).
- Muy buena relacion de compresion y performance de acceso.
- Se utiliza en muchas distribuciones liveCDs y liveUSB.
- Soporta compresión LZO para mejor performance en sistemas embebidos (menor relación de compresion).

Block fileSystems

- LZO es adecuada cuando se tiene poco poder de procesamiento (CPU lenta).
- **squashfs** también soporta algoritmo XZ para una mejor compresión, a costa de mayor utilización de CPU.
- Benchmarks: aproximadamente 3 veces menor a un filesystem ext3 y entre 2 y 4 veces más rápido (link a comparaciones).

Block fileSystems

- Para utilizar **squashfs** se debe instalar el paquete **squashfs-tools**.
- Para crear una imagen, se utiliza el comando **mksquashfs**
- Ejemplo: **mksquashfs rootfs/ rootfs.sqfs**
- ATENCIÓN: si la imagen ya existe, debe removerse primero o utilizar la opción **-noappend**.
- Para instalar la imagen se copia directamente el archivo mediante el comando **dd**.

Block fileSystems

- EJEMPLO: Asumiendo que la partición donde queremos copiar la imagen es **/dev/sc1**.
  dd if=rootfs.sqfs of=/dev/sdc1
- Montar el filesystem:
  mount -t squashfs /dev/sdc1 /mnt/root

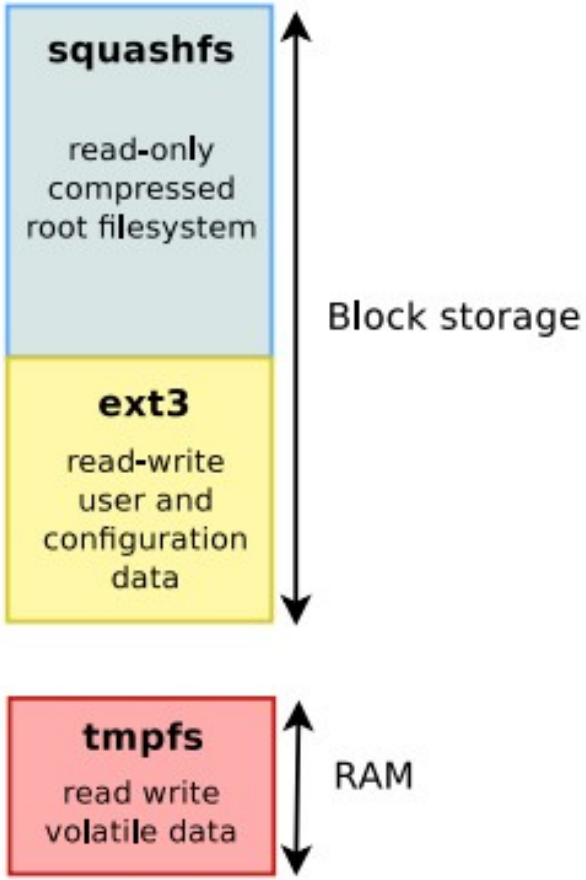
Block fileSystems

- Mencionamos un filesystem útil: **tmpfs**.
- No es un block filesystem: no define una estructura en disco.
- Es un pseudo filesystem.
- Se utiliza para almacenar datos volátiles en RAM.
- Lo mencionamos porque almacena información en RAM como un ramdisk (que si es block fs).
- Ejemplo: **mount -t tmpfs varrun /var/run**.

Block fileSystems

- Con lo visto, es buena idea separar el almacenamiento tipo block en varias partes.
- Una partición de solo lectura comprimida (**squashfs**) utilizada para binarios y kernel.
  - Ventajas: la compresión ahorra espacio y ser solo-lectura protege el sistema de errores o corrupción de datos.
- Una partición de lectura-escritura con un journaled filesystem (**ext3** o **ext4**) para datos de usuario y configuraciones.
- Almacenamiento en RAM para datos volátiles (**tmpfs**).

Block fileSystems



Gracias.