Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

File Systems

Overview

Hes·so FRIBOURG FREIBURG
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

References

[1]: <Linux Kernel sources>/Documentation/filesystems

[2]: http://www.tldp.org/HOWTO/html_single/SquashFS-HOWTO

[3]: http://squashfs.sourceforge.net

[4]: tree.celinuxforum.org/CelfPubWiki/ELCEurope2008Presentations?action=AttachFile&do=get&target=squashfs-elce.pdf

[5]: http://superuser.com/questions/228657/which-linux-filesystem-works-best-with-ssd //File for SSD card

[6]: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/index.html // very good site

[7]: https://code.google.com/p/cryptsetup/

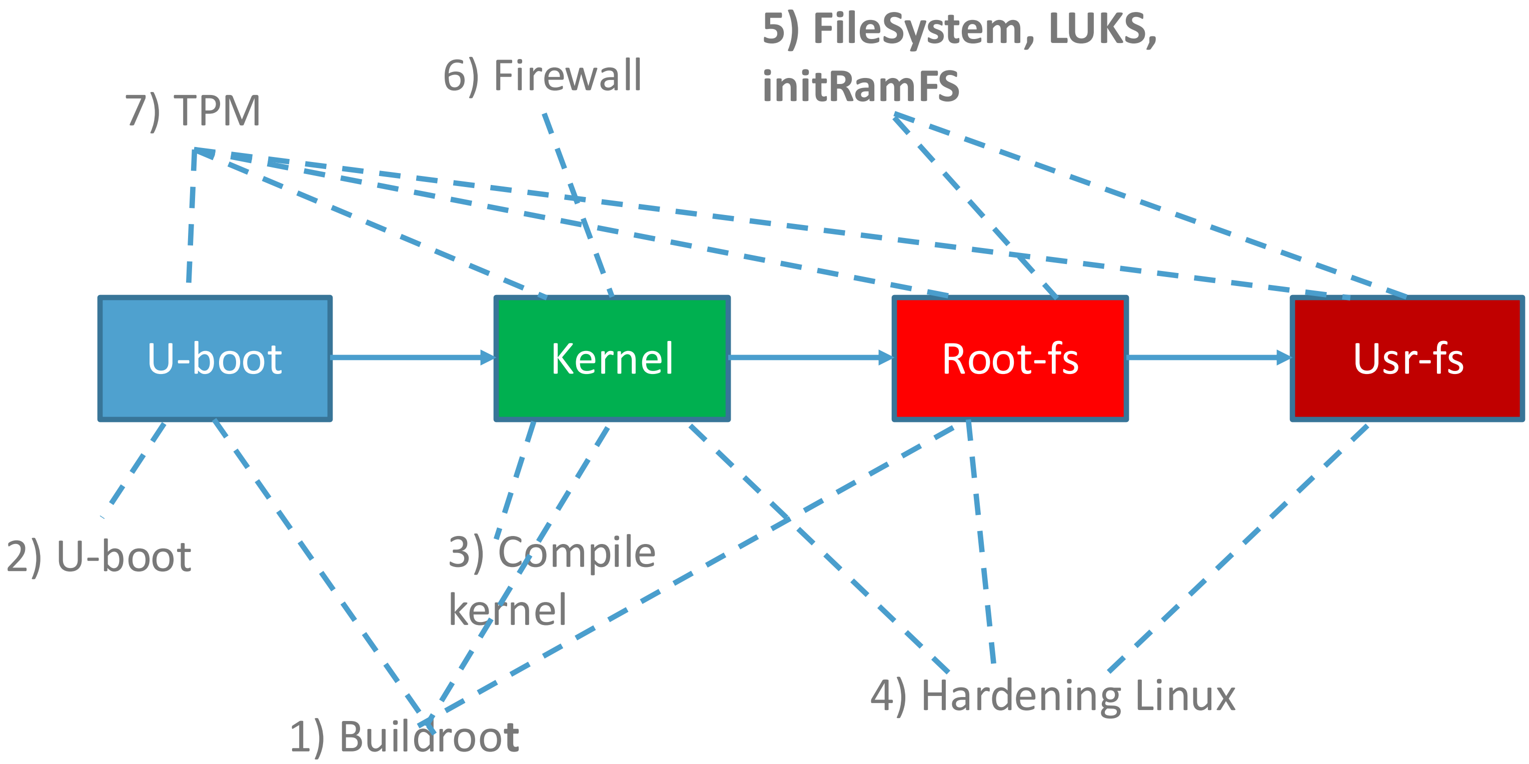
Power off embedded FS

[8]: http://stackoverflow.com/questions/14460091/embedded-file-system-and-power-off

[9]: https://elinux.org/images/0/02/Filesystem_Considerations_for_Embedded_Devices.pdf

Course summary, main chapters

1) Buildroot
2) U-boot
3) Compile kernel
4) Hardening Linux
5) **FileSystem, LUKS, InitRamFS**
6) Firewall
7) TPM (Trusted Plarform Module)

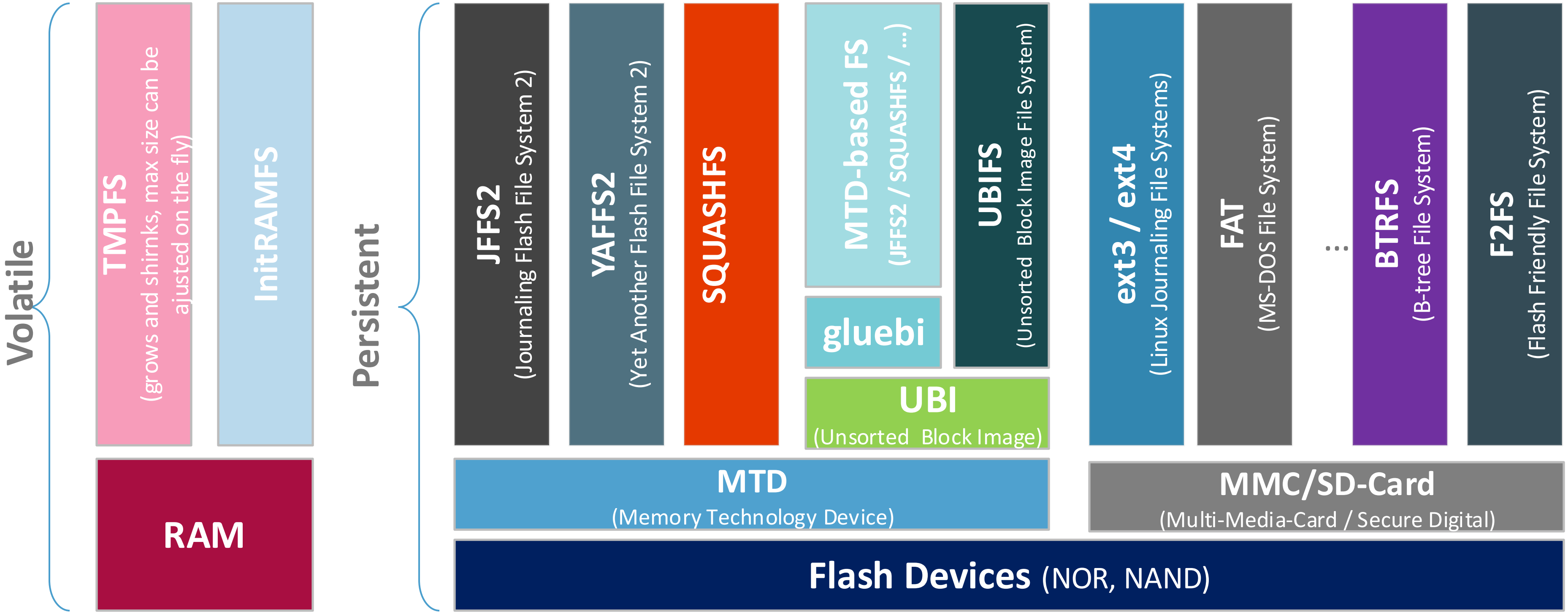


File system summary

- Flash memory technologies: MTD, MMC
- Journalized file systems: ext4, …
- B-Tree file systems: BTRFS, …
- Log file systems: F2FS, …
- ext4 main commands
- /etc/fstab
- Squash file system
- Tpmfs, devtpmfs

Systèmes de fichiers (Cours CSEL)

- For embedded systems, there are two categories of file systems: **volatile RAM-based** systems and **persistent Flash-based systems** (NOR and, NAND technology)
- Two main technologies are available for Flash: **MTD** (Memory Technology Device) and **MMC/SD-Card** (Multi-Media-Card / Secure Digital Card)
- There are a multitude of file systems for each of these technologies



Volatile

TMPFS (grows and shirnks, max size can be ajusted on the fly)

InitRAMFS

RAM

Persistent

JFFS2 (Journaling Flash File System 2)

YAFFS2 (Yet Another Flash File System 2)

SQUASHFS

MTD-based FS (JFFS2 / SQUASHFS / …)

gluebi

UBIFS (Unsorted Block Image File System)

UBI (Unsorted Block Image)

MTD (Memory Technology Device)

ext3 / ext4 (Linux Journaling File Systems)

FAT (MS-DOS File System)

…

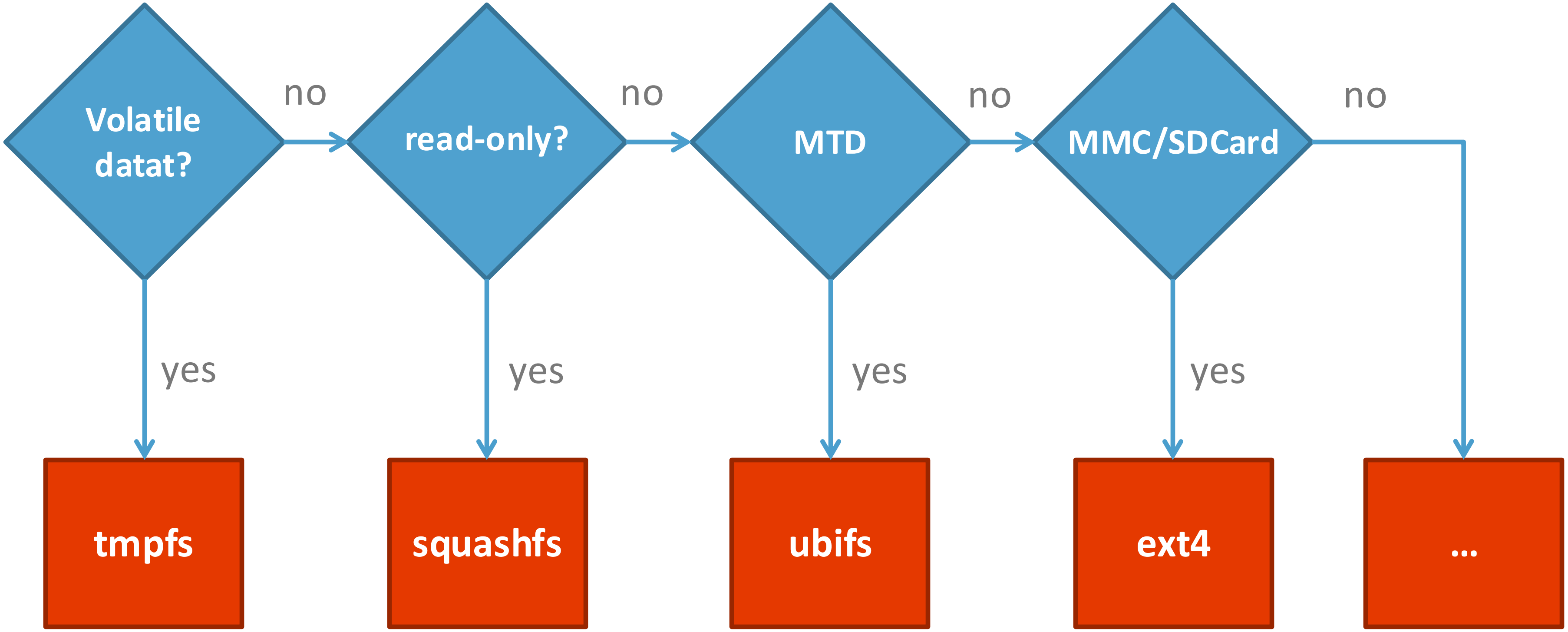
BTRFS (B-tree File System)

F2FS (Flash Friendly File System)

MMC/SD-Card (Multi-Media-Card / Secure Digital)

Flash Devices (NOR, NAND)

Choosing a File System



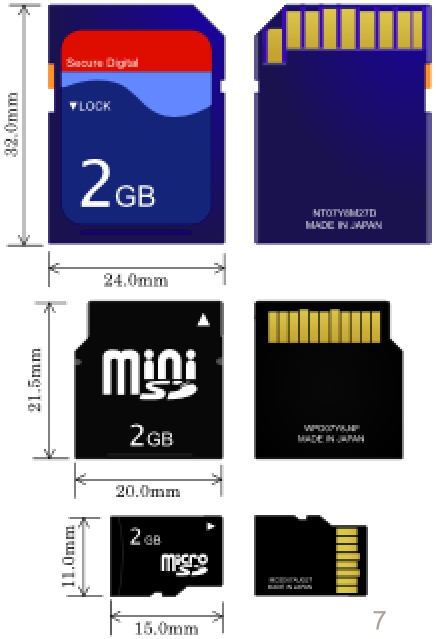
See the Linux kernel documentation for more details on file systems
Documentation/filesystems/

MMC technologies [9]

- **MMC**: MultiMediaCard is a memory card unveiled in 1997 by SanDisk and Siemens based on NAND flash memory

- **eMMC**: embedded MMC is just a regular MMC in a BGA package, that is welded to the platform

- **SD Card**: SecureDigital Card was introduced in 1999 based on MMC but adding extra features such as security

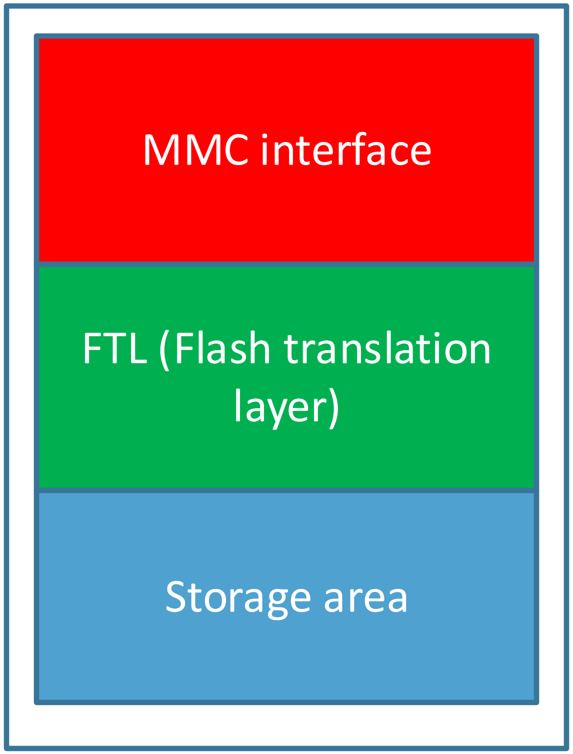


Pictures: Wikipedia

Inside MMC technologies [9]

MMC – eMMC - SD Card are composed by 3 elements:

- MMC interface: handle communication with host
- FTL (Flash translation layer):
- Storage area: array of NAND chips



MMC interface

FTL (Flash translation layer)

Storage area

FTL is a small controller running a firmware. Its main purpose is to transform logical sector addressing into NAND addressing.

It also handles:

- Bad block management
- Garbage collection
- Wear levelling (frequently modified blocks are mapped to different portions of memory)

MMC/SD-Card partition alignment

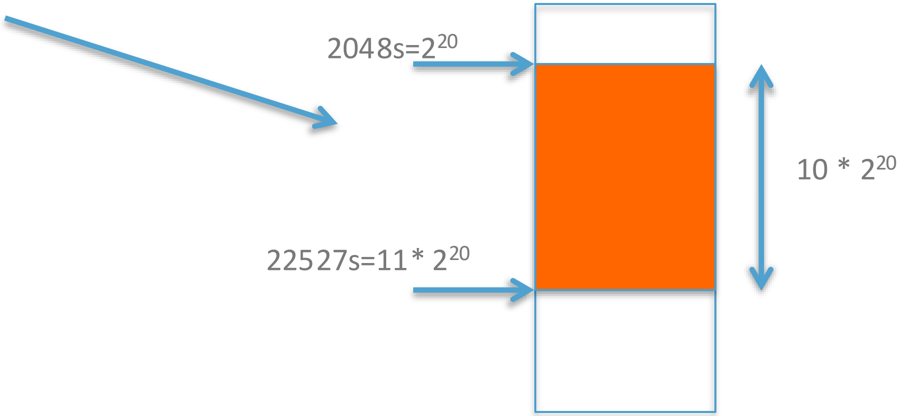
- Partition alignment is critical for MMC/SD-Card as, being memory-based devices, data is written and read in blocks known as pages. When partitions aren't aligned, the block size of filesystem writes isn't aligned to the block size of the MMC/SD-Card

- To easily guarantee proper data alignment, **the starting sector of each partition must be a multiple of 2^{20}** (= 1'048'576) Bytes

- E.g. for sdb (the sector size is generally 512 bytes)
  - `# sudo fdisk /dev/sdb` //press n for a new partition, p for primary and enter a start sector of at least 2048 (2048 * 512 = 1'048'576)

or

  - `sudo parted /dev/sdb mkpart primary ext4 2048s 22527s` // s=1 sector = 512 bytes



File systems, embedded systems

Embedded Systems can use different filesystems

- ext4, ext3, ext2
- BTRFS
- F2FS
- NILFS2
- XFS
- (ZFS)

- Squashfs

MMC, SD-Card

- Tmpfs, InitRAMFS

RAM

- UBIFS
- JFFS2
- YAFFS2

MTD (Memory Technology Device)

Journalized file system [9]

A journalized filesystem keeps track of every modification in a journal in a dedicated area.

- Journal allows to restore a corrupted filesystem
- Modifications are first recorded in the journal
- Modifications are applied on the disk
- If a corruption occurs: The File System will either keep or drop the modifications
  - Journal is consistent: we replay the journal at mount time
  - Journal is not consistent: we drop the modifications

Well known journalized filesystems:

- EXT3, EXT4
- XFS
- Reiser4

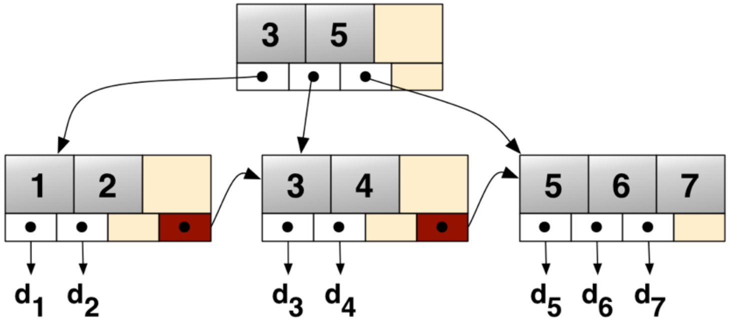
B-TREE/CoW [9]

- B+ tree is a data structure that generalized binary trees (see also https://en.wikipedia.org/wiki/B%2B_tree)

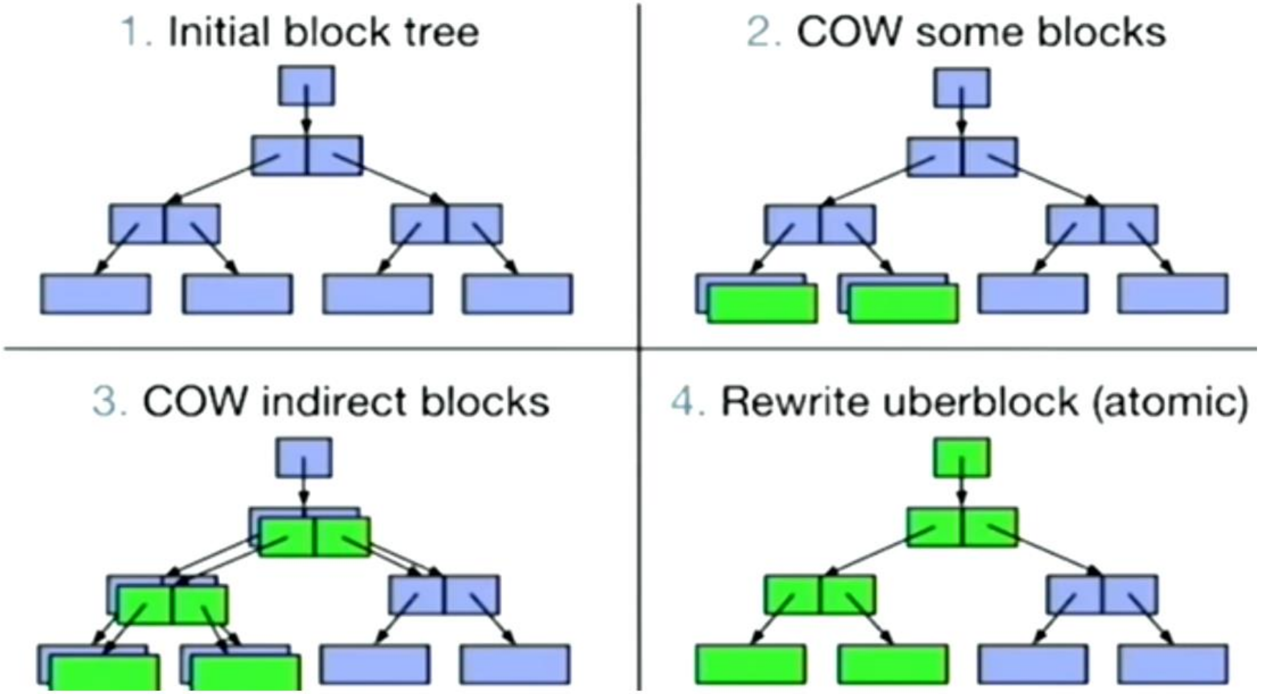
- CoW (Copy on Write) is used to ensure no corruption occurs at runtime:
  - The original storage is never modified. When a write request is made, data is written to a new storage area
  - Original storage is preserved until modifications are committed
  - If an interruption occurs during writing the new storage area, original storage can be used.

Well known filesystems using CoW:

- ZFS
- BTRFS
- NILFS2



B-tree filesystem [9]



Log filesystem [9]

Log-structured filesystems use the storage medium as circular buffer and new blocks are always written to the end.

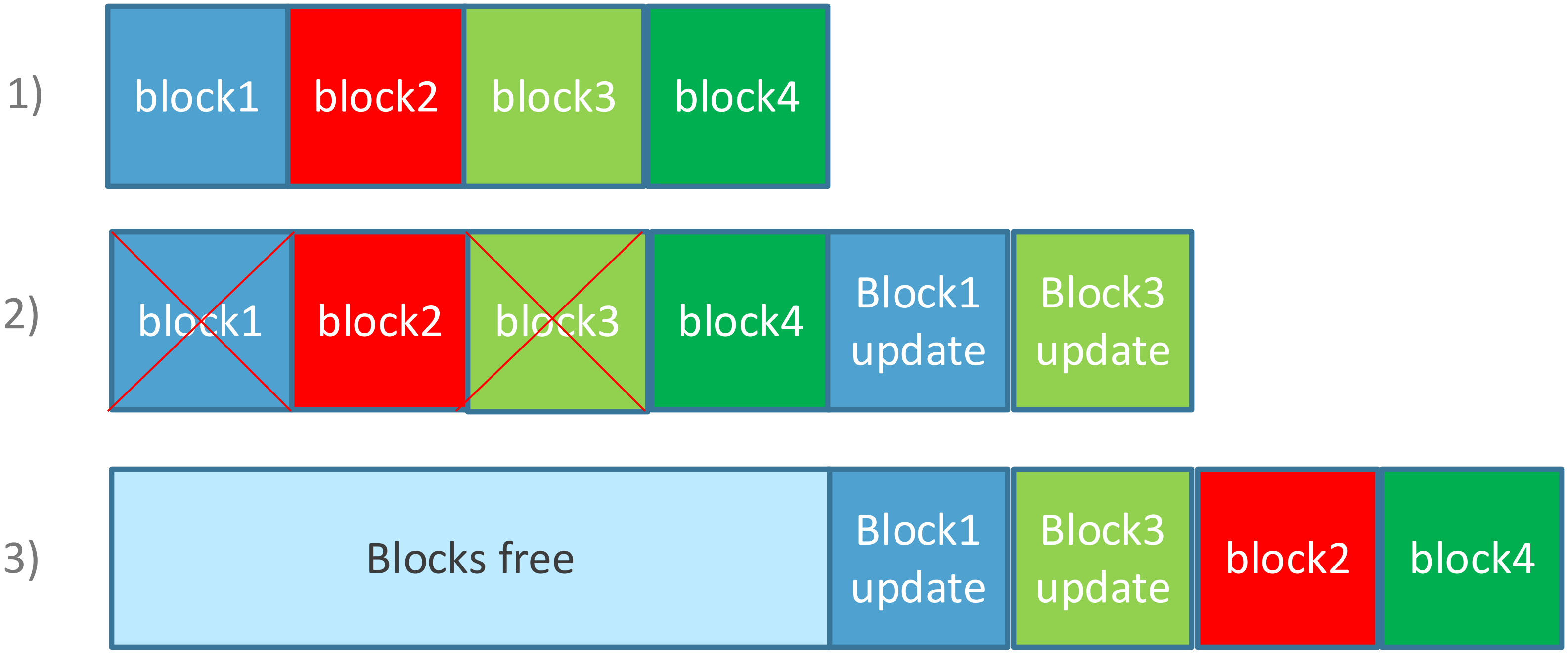
- Log-structured filesystems are often used for flash media since they will naturally perform **wear-levelling**

- The log-structured approach is a specific form of copy-on-write behavior

Well known log filesystems:

- F2FS
- NILFS2
- JFFS2
- UBIFS

Log filesystem [9]



1) block1 block2 block3 block4

2) block1 block2 block3 block4 Block1 update Block3 update

3) Blocks free Block1 update Block3 update block2 block4

1) Initial state
2) Block1 & block3 are updated, old block1 and block3 are not used anymore
3) Garbage copies block2 and block4, and frees old block1,2,3,4

A few examples of the different types of filesystems

BTRFS (B-Tree filesystem) [9]

- BTRFS is a “new” file system compared to EXT. It is originally created by Oracle in 2007. it is a B-Tree filesystem.
- It is considered stable since 2014
- Since 2015 BTRFS is the default rootfs for openSUSE.
- BTRFS was inspired from both Reiserfs and ZFS.
- Theodore Ts’o (ext3-ext4 main developer) said that BTRFS has a better direction than ext4 because: *it offers improvements in scalability, reliability, and ease of management*

BTRFS (B-Tree filesystem)

On PC:

Create new partition with fdisk or parted commands

```
sudo dnf install btrfs-progs.x86_64      // install tools for btrfs (Fedora)
sudo apt install btrfs-progs             //Ubuntu
sudo mkfs.btrfs /dev/sdb3                       // format partition 3
sudo btrfs filesystem label /deb/sdb3 usr_btrfs
```

On NanoPi

```
mount /dev/mmcblk0p3 -t btrfs /mnt
```

- or with /etc/fstab

```
#/dev/mmcblk0p3 /mnt      btrfs    defaults       0       0
```

F2FS (Flash-Friendly File System) [9]

- It is a log filesystem. It can be tuned using many parameters to allow best handling on different supports

- F2FS features:
  - Atomic operations
  - Defragmentation
  - TRIM support (reporting free blocks for reuse)

F2FS (log FileSystem)

On PC

Create new partition with fdisk or parted commands

- `sudo dnf install f2fs-tools.x86_64` // install tools for f2fs (Fedora)
- `sudo apt install f2fs-tools` // install tools for f2fs (Ubuntu)
- `sudo mkfs.f2fs -l usr_f2fs /dev/sdb3` // format partition 3

On NanoPi

- `mount /dev/mmcblk0p3 -t f2fs /mnt`
- `or with /etc/fstab`
  - `#/dev/mmcblk0p3 /mnt      f2fs    defaults      0      0`

XFS [9]

XFS was developed by SGI in 1993.

- Added to Linux kernel in 2001
- On disk format updated in Linux version 3.10
- XFS is a journaling filesystem.
- Supports **huge** filesystems
- Designed for scalability
- Does not seem to be handling power loss (standby state) well

XFS

On PC

Create new partition with fdisk or parted commands

- `sudo dnf install xfsprogs.x86_64` // install tools for btrfs (Fedora)
- `sudo apt install xfsprogs` // install tools for btrfs (Ubuntu)
- `sudo mkfs.xfs  -L usr_xfs /dev/sdb3` // format partition 3

On NanoPi

- `mount /dev/mmcblk0p4 -t xfs /mnt/xfs/`
- `or with /etc/fstab`
  - `#/dev/mmcblk0p3 /mnt        xfs    defaults        0        0`

NILFS2 (New Implementation of a Log-structured File System) [9]

- Developed by Nippon Telegraph and Telephone Corporation
- NILFS2 Merged in Linux kernel version 2.6.30
- NILFS2 uses log filesystem and B-Tree technologies
- Userspace garbage collector

NILFS2 (New Implementation of a Log-structured File System)

On PC

Create new partition with fdisk or parted commands

- `sudo dnf install nilfs-utils.x86_64` // install tools for nilfs (Fedora)
- `sudo apt install nilfs-tools` // install tools for nilfs (Ubuntu)
- `sudo mkfs.nilfs2 -L usr_nilfs /dev/sdb3` // format partition 3

On NanoPi

- `mount /dev/mmcblk0p3 -t nilfs2 /mnt`
- `or with /etc/fstab`
  - `#/dev/mmcblk0p3 /mnt        nilfs2    defaults        0        0`

ZFS (Zettabyte (10^{21})File System) [9]

ZFS is a combined file system and logical volume manager designed by Sun Microsystems

- ZFS is a B-Tree file system
- Provides strong data integrity
- Supports huge filesystems
- **Not intended for embedded systems** (requires a lot of RAM)

Ext2, ext3, ext4 file system [9]

[9]: “Filesystem considerations for embedded devices” is a good study about filesystems used on embedded systems
https://elinux.org/images/0/02/Filesystem_Considerations_for_Embedded_Devices.pdf

These files systems are very used in different Linux distribution

- EXT filesystem was created in April 1992
- EXT2 replaced it in 1993
- EXT3 evolution added a **journal** and was merged in 2001
- EXT4 arrived as a stable version in the Linux kernel in 2008

Conclusions [9]

Performances:

- EXT4 is currently the best solution for embedded systems using MMC
- F2FS and NILFS2 show very good write performances

Features:

- BTRFS is a next generation filesystem
- NILFS2 provides simpler but similar features

Scalability:

- EXT4 clearly doesn’t scale as well as BTRFS and F2FS