Filesystems

Dr Peadar Grant

February 14, 2024

Contents

1 Disk-based filesystems

Disk-based file systems interposes between the application layer (including the user-space OS tools like File Manager) and the block devices. A file system is created on a block device (= a block device is formatted with a file system) that is then mounted by the host's operating system.

1.1 Common filesystems

As of the present time the most common disk-based filesystems are:

Microsoft Windows: normally NT File System (NTFS) for fixed disks.

Apple Mac OS X: APFS (solid-state disks only), HFS+ (formerly all fixed media, remains on magnetic disks only)

Linux-based systems commonly use ext4, but a large range of other systems commonly appear: ext3, xfs, reiserfs, ZFS*

UNIX-based systems tend to have an OS-specific filesystem w/ similar behaviour to linux, often ZFS

Wikipedia has collated a comparison of file systems.

1.2 Roles of filesystems

Space management using allocation table. Dealing with *fragmentation*.

Naming possibly case (sensitive | insensitive | preserving)

Hierarchy using directories.

- Historically and some specialist systems: **flat** filesystem

Metadata such as file creation / modification time

Permissions management via metadata to allow (read | write | execute) to specific users.

- Common permissions are owner / group / world (others) in UNIX.
- The host OS, not the filesystem, actually enforces these restrictions.

Maintaining integrity using redundancy, checksums.

Durable storage by journalling

2 Journalling

Durability is the key expectation:

- data once written to filesystem isn't lost (within reasonable assumptions).

Journalling is where the intent to change data is recorded to the journal before the actual data itself is modified:.

1. Data is written to the on-disk journal
2. The actual on-disk file data is changed
3. The journal is deleted

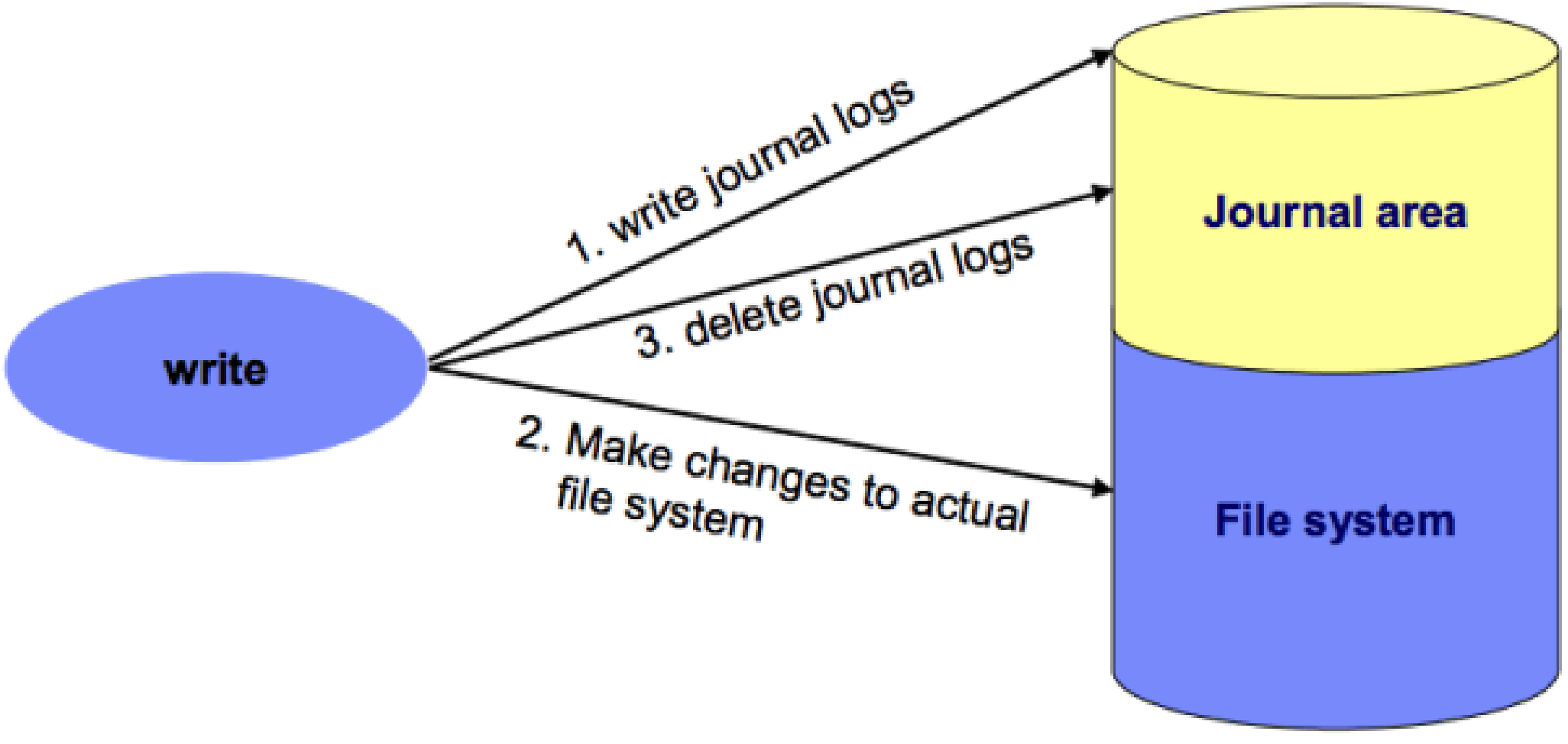

Figure 1: Journalling filesystem

2.1 Alternatives to Journalling

Copy-on-write: where a modification is first written and then the original deleted

Log-structured file system: the journal itself is the filesystem

3 Single vs multi-root

Although more an OS-characteristic, there is an important difference between single and multi-root operating systems that becomes relevant for discussion of filesystems:

Single root operating systems

- boot with **one** filesystem mounted as the **root** `/` directory.
- All others are attached at *mountpoints* within the root directory.

Multi-root operating system has multiple roots:

- often indicated by drive-letters e.g. `C:`, `D:`.
- other OSes (e.g. VMS, Novell) use volume labels (e.g. `SYS`)

In general, most UNIX-like OS are single root. Windows is multi-root.

3.1 Linux file system operations

Linux and UNIX tend to use similar commands to those shown here:

- Always check the precise requirement for your OS.

3.2 Creation

File system creation, also called Formatting, is done through the `mkfs` command:

```
# review help text and manpage
mkfs -h
man mkfs

# check where mkfs is located (usually /sbin)
which mkfs

# from that, find the available filesystems
ls -l /sbin | grep mkfs

# creating a filesystem (aka formatting)
# e.g. make an ext4 on /dev/sdb1
mkfs -t ext4 /dev/sdb1
```

3.3 Mounting

Mounting is where a particular a block device formatted with a particular filesystem is attached into the file system hierarchy:

- A folder, called the “mount point” is created on an existing (usually the root) filesystem.
- Mounting the additional drive then “covers up” that folder with root of the mounted drive.

Consider the example of mounting a block device `/dev/sdb1` at the mountpoint `/mnt/data`:

```
# review mount command
mount -h #inbuilt help
man mount #man page

# create the mountpoint (-p allows it to already exist)
mkdir -p /mnt/data

# issue the mount}
mount -t ext4 /dev/sdb1 /mnt/data

# viewing mounted systems
mount
```

3.4 Mounting on boot

The manual method has some shortcomings:

- Manual mounts / unmounts persist only as long as the system is running.

Instead we'd like to have a configuration file of mountable filesystems.

The `/etc/fstab` file provide this:

- lists filesystems that can be mounted / unmounted without specifying the block device to the `mount` command
- additionally can indicate these should be mounted automatically when the system starts up.

3.5 /etc/fstab format

The `/etc/fstab` file format is whitespace separated of the following fields:

1. `device-spec` of the device to mount (device name, label, UUID, other means to idenitfy)
2. `mount-point` where filesystem can be accessed once mounted
3. `fs-type` identifying the filesystem
4. `options` that may be needed: `defaults` per-filesystem, `noauto` to not mount on boot
5. `dump` - for use by the `dump` program for backups (backups to be discussed later)
6. `pass` - order for `fsck` to check errors at boot time: 0=don't, 1=during, 2=after

More advanced generic example, on Wikipedia:

```
# device-spec   mount-point     fs-type      options
LABEL=/         /               ext4         defaults
/dev/sda6       none            swap         defaults
none            /dev/pts        devpts       gid=5,mode=620
none            /proc           proc         defaults
none            /dev/shm        tmpfs        defaults

# Removable media
/dev/cdrom      /mnt/cdrom      udf,iso9660  noauto,owner,ro

# NTFS Windows 7 partition
/dev/sda1       /mnt/Windows    ntfs-3g      quiet,defaults,locale=en_US.utf8,umask=0,r

# Partition shared by Windows and Linux
/dev/sda7       /mnt/shared     vfat         umask=000
```

```
# Mounting tmpfs
tmpfs           /mnt/tmpfschk   tmpfs        size=100m

# Mounting cifs
//cifs_server_name/ashare  /store/pingu    cifs         credentials=/root/smbpass.txt

# Mounting NFS
nfs_server_name:/store    /store          nfs          rw
```

Simpler sample of `/etc/fstab` on a Raspberry-Pi:

```
proc            /proc           proc    defaults          0       0
PARTUUID=af832fc2-01  /boot           vfat    defaults          0       2
PARTUUID=af832fc2-02  /               ext4    defaults,noatime  0       1
/dev/sdb2             /data           ext4    defaults,noatime  0       0
# a swapfile is not a swap partition, no line here
```

```
#   use  dphys-swapfile swap[on|off]  for that
```

Can often also identify block devices by other means than their `/dev` node, for example partition UUID. Avoids ambiguities if drives are swapped around. Check help for `mount` command to be sure.

3.5.1 Automounters

Other means to identify and mount filesystems exist on Linux (particularly) nowadays: systemd unit files, pmount, automount.

3.6 Windows

3.6.1 Creation

By demonstration through Computer Management.

3.6.2 Mounting

Generally a drive letter will be assigned.

Windows can also do a UNIX-style mount to a folder within another filesystem. Not commonly used though.

4 Virtual filesystems

A filesystem as such is implemented within the operating system normally in the form of code. This may in fact allow filesystems to be created that do not directly reside on any block device.

Network filesystem where the host connects to a filesystem located on another computer using file-sharing protocols like SMB, NFS.

Object storage filesystems where the host connects to a storage service that does not itself implement the normal hierarchical filesystem, but can appear to do so:

- Software to map WebDAV and systems like S3

Device filesystem usually mounted at `/dev` used in UNIX-like OSes to provide access and permission control to peripheral devices (e.g. HID, serial, LPT, block devices).

Special filesystems seen in UNIX:

- `sysfs` and `configfs` kernel

- `/proc` within info on processes