

Linux 分区格式、文件系统 与文件类型全解析

一次搞懂三个概念的层层关系

适用于系统运维 · 服务器维护 · Linux 学习

目录

引言 一张图看懂三者的关系

第一部分 Linux 分区格式（分区表）

- 一、什么是分区表
- 二、MBR 主引导记录
- 三、GPT 分区表
- 四、MBR 与 GPT 对比
- 五、分区类型：主分区 / 扩展分区 / 逻辑分区
- 六、LVM 逻辑卷管理

第二部分 Linux 文件系统（文件格式）

- 一、什么是文件系统
- 二、文件系统的三层架构
- 三、ext 家族：ext2 / ext3 / ext4
- 四、XFS
- 五、Btrfs
- 六、ZFS 与其他常用文件系统
- 七、特殊文件系统：swap / tmpfs / 虚拟文件系统
- 八、核心概念：inode 与日志
- 九、主流文件系统对比与选型

第三部分 Linux 文件类型

一、核心哲学：一切皆文件

二、七种文件类型详解

三、如何识别文件类型

四、硬链接与软链接

第四部分 总结：三者区别一目了然

一、三个概念的分层总结

二、常见误区辨析

三、实操速查

引言 一张图看懂三者的关系

很多刚接触 Linux 的人会被三个概念搞混：分区格式（partition format）、文件系统（filesystem）、文件类型（file type）。它们听起来都跟"文件"有关，实际上却处于完全不同的层次。理解这三者的关系，是掌握 Linux 存储体系的第一步。

用一个比喻来理解：把一块硬盘想象成一栋写字楼——

- 分区格式（分区表）= 楼层的划分图纸，决定这栋楼隔成几间、每间多大、怎么编号；
- 文件系统 = 每间办公室内部的装修布局，决定办公桌（文件）怎么摆放、怎么记录；
- 文件类型 = 办公室里摆的到底是什么东西，是文件柜（目录）、快捷方式（软链接），还是电话线（管道）。

技术上的层次关系如下：

```
物理硬盘 (/dev/sda)
| ① 分区 (Partition) —— 由分区表管理：MBR / GPT
▼
分区 (/dev/sda1, /dev/sda2 ...)
| ② 格式化 (Format) —— 写入文件系统：ext4 / XFS / Btrfs ...
▼
文件系统 (Filesystem)
| ③ 组织文件—— 文件 / 目录 / 链接 / 设备节点 ...
▼
文件 (File) , 由 inode 与目录项管理
```

【一句话记忆】

分区表决定"盘怎么切" → 文件系统决定"切出来的空间怎么存" → 文件类型决定"存的都是什么"。三者层层嵌套，缺一不可。

第一部分 Linux 分区格式（分区表）

一、什么是分区表

分区表（Partition Table）是磁盘上一个记录分区信息的数据结构，它告诉操作系统：这块磁盘被划分成了几个分区、每个分区的起始位置和大小。没有分区表的磁盘只是一块"无差别的存储块"。

可以把分区表理解为磁盘的"目录"或"地图"。目前主流的两两种分区表格式是 MBR 和 GPT。

在 Linux 中，磁盘设备通过块设备文件访问，常见命名规则：

- /dev/sda：第一个 SCSI/SATA 磁盘（sda→sdb→sdc...）
- /dev/nvme0n1：第一个 NVMe 控制器上的第一个命名空间
- /dev/vda：VirtIO 虚拟磁盘（云主机常见）
- 分区命名：/dev/sda1、/dev/nvme0n1p2（设备名以数字结尾时用 p 分隔）

二、MBR 主引导记录

MBR（Master Boot Record，主引导记录）是上世纪 80 年代 IBM PC 时代诞生的经典分区表格式，兼容性极强，几乎所有系统都能识别。

结构特点

- MBR 位于磁盘的第一个扇区（512 字节），其中引导代码占 440 字节，分区表只占 64 字节
- 每个分区信息需要 16 字节，因此最多支持 4 个主分区
- 分区表只有一份，没有冗余备份，损坏后难以恢复
- 使用 32 位扇区寻址，最大只支持 2 TB 的磁盘

主要限制

- 最大磁盘容量：2 TB（超过部分无法识别）
- 最多 4 个主分区，或 3 个主分区 + 1 个扩展分区（扩展分区内可再划分多个逻辑分区）
- 无分区校验（CRC），数据损坏时无感知
- 适用于传统 BIOS 启动模式

【提示】

如今 2TB 以上的大容量硬盘已成为主流，MBR 的 2TB 限制使其逐步被 GPT 取代。仅在极老旧的设备、嵌入式系统或特殊兼容需求下才继续使用 MBR。

三、GPT 分区表

GPT（GUID Partition Table，全局唯一标识分区表）是 UEFI 时代的新一代分区表格式，是现代磁盘的标准选择。

结构特点

- 磁盘开头保留一份“保护性 MBR”（Protective MBR），防止不认识 GPT 的旧系统误操作
- 主分区表与备份分区表各一份，分区表在磁盘末尾有完整备份，容错率高
- 每个分区有唯一的 GUID 标识，支持自定义分区名称
- 分区表自带 CRC32 校验，可检测数据损坏

主要优势

- 支持最大 9.4 ZB 的磁盘（远超当前硬件水平）
- 默认支持 128 个分区（可通过修改分区表头扩展）
- 原生支持 UEFI 启动与 Secure Boot

- 分区表多份冗余 + CRC 校验，数据更安全

使用 GPT 的 UEFI 启动方式需要专门的 EFI 系统分区（ESP），该分区必须格式化为 FAT32，用于存放引导程序。

四、MBR 与 GPT 对比

对比项	MBR	GPT
最大磁盘容量	2 TB	约 9.4 ZB（理论）
分区数量	最多 4 个主分区（或 3 主 + 1 扩展）	默认 128 个，可扩展
分区表备份	无（仅一份）	磁盘首尾各一份
数据校验	无	CRC32 校验
启动方式	BIOS（传统）	UEFI（现代）
大容量支持	不支持 >2TB	原生支持
适用场景	老旧设备 / 特殊兼容需求	现代服务器、桌面、SSD
常用工具	fdisk	gdisk / parted

【结论】

新设备优先选择 GPT，尤其是 SSD、大容量硬盘或需要 UEFI 启动的场景。GPT 兼容 MBR 时代的引导流程，双保险设计更可靠。

五、分区类型：主分区 / 扩展分区 / 逻辑分区

分区类型	说明	限制
主分区	可直接使用的独立分区，包含操作系统引导信息	MBR 下最多 4 个；GPT 下无数量限制
扩展分区	MBR 专有概念，本身不能直接使用，用于突破 4 主分区限制	只能有 1 个
逻辑分区	位于扩展分区内部，可正常格式化使用	MBR 下最多 15 个
EFI 系统分区	UEFI 引导专用分区	须为 FAT32，存放引导程序

六、LVM 逻辑卷管理

在 MBR/GPT 划分的物理分区之上，Linux 提供 LVM（Logical Volume Manager，逻辑卷管理）技术，这是服务器存储管理的"灵魂"。

- 物理卷（PV）：将物理分区或整块磁盘格式化为 LVM 可识别的格式
- 卷组（VG）：将一个或多个 PV 合并成一个大的存储池
- 逻辑卷（LV）：从 VG 中划分出的逻辑空间，最终格式化为文件系统供系统使用

LVM 的最大优势是动态扩展：当逻辑卷空间不足时，无需重启服务器，只需将新磁盘加入卷组再扩展逻辑卷即可，数据零丢失、业务无感知。

```
# 常用分区命令

lsblk                    # 查看块设备树状结构
lsblk -f                 # 显示文件系统类型和 UUID
sudo fdisk -l /dev/sdb   # 查看分区表详情
sudo fdisk /dev/sdb      # 交互式 MBR 分区
sudo gdisk /dev/sdb      # 交互式 GPT 分区
sudo parted /dev/sdb mklabel gpt          # 创建 GPT 分区表
sudo parted /dev/sdb mkpart primary ext4 1MiB 50GiB # 创建分区
sudo growpart /dev/sda 1                  # 扩展分区到可用空间
sudo blkid                               # 查看所有设备的 UUID 和文件系统类型
```

第二部分 Linux 文件系统（文件格式）

一、什么是文件系统

文件系统（Filesystem）是操作系统在存储设备上组织、管理和存储数据的方法。它定义了文件如何命名、如何存放、如何检索、如何记录元数据（权限、时间戳、属主等），以及如何保证数据完整性。

分区（或逻辑卷）创建之后，必须"格式化"（写入某种文件系统）才能存储文件。格式化的过程相当于在分区内建立文件系统的数据结构（超级块、inode 表、数据块等）。



二、文件系统的三层架构

Linux 文件系统采用三层架构设计，这也是它能同时支持几十种文件系统的原因：

层次	作用	说明
逻辑文件系统	面向应用的 API 层	管理打开文件表、文件描述符
虚拟文件系统（VFS）	抽象层	为不同文件系统提供统一接口，让用户"感觉不到"底层差异
物理文件系统	具体实现层	在磁盘上实现具体的数据组织（ext4、XFS、Btrfs...）

正是因为 VFS 的存在，你可以同时使用 ext4（根分区）、XFS（数据盘）、vfat（U 盘）而无需关心差异，它们都表现为统一的"文件"接口。

三、ext 家族：ext2 / ext3 / ext4

ext（Extended File System，扩展文件系统）是 Linux 最传统的文件系统家族，ext4 是目前绝大多数 Linux 发行版的默认选择。

ext2

- 最早进入 Linux 的成熟文件系统，结构简单、运行稳定
- 无日志功能 (journaling) ，断电或崩溃后需要长时间 fsck 检查
- 最大文件 2 TB、最大分区 4 TB
- 如今仅用于小分区、嵌入式设备或需要最小写入的场景

ext3

- 在 ext2 基础上增加日志功能，崩溃后快速恢复
- 可与 ext2 互相升级/降级，无需重新格式化
- 最大文件 2 TB、最大分区 32 TB

ext4

- 2008 年随 Linux 2.6.28 发布，当前最广泛使用的文件系统
- 支持最大文件 16 TB、最大分区 1 EB
- 引入 extents (区段) 机制，大幅减少大文件碎片
- 延迟分配 (delayed allocation) 提升写入性能
- 支持在线扩容与缩容 (resize2fs)
- 向后兼容 ext2/ext3，可原地升级
- 缺点：不支持快照、不支持透明压缩、无数据校验

【定位】

ext4 是"稳妥之选"：成熟、稳定、兼容性最好，适合绝大多数通用场景——桌面系统、普通服务器、系统盘、中小数据库。

四、XFS

XFS 由 SGI 于 1993 年为 IRIX 系统开发，2001 年移植到 Linux，是 RHEL/CentOS/Rocky 等红帽系发行版的默认文件系统。

核心特点

- 64 位高性能日志文件系统，专为大文件、高并发 I/O 设计
- 支持最大文件 8 EB、最大分区 8 EB，可轻松管理 PB 级存储
- 采用分配组（Allocation Group）设计，多核并行 I/O 能力极强
- 支持在线扩容（xfs_growfs），无需卸载
- 元数据日志，崩溃后恢复速度快

主要限制

- 只支持扩容，不支持缩容（这是选型时的重要考量）
- 大量小文件场景下元数据开销较大
- 无快照、无压缩功能

【经验提醒】

xfs_growfs 作用于挂载点而不是设备：应使用 xfs_growfs /data，而不是 xfs_growfs /dev/sdb1。ext4 的 resize2fs 则作用于设备。这是运维面试和实操中常见的“坑”。

五、Btrfs

Btrfs（B-Tree Filesystem）由 Oracle 于 2007 年发起，是一个面向未来的现代文件系统，openSUSE 和 Fedora 的默认文件系统。

核心特点

- 写时复制 (Copy-on-Write, CoW) : 修改数据时先复制再写入, 天然支持快照
- 内置快照与子卷 (subvolume) 管理, 秒级快照、方便回滚
- 透明压缩: 支持 zlib / lzo / zstd 压缩, 节省空间
- 数据校验和 (checksum) + 自愈能力, 可检测并修复静默数据损坏
- 内置 RAID 0/1/10 (RAID 5/6 仍标记为实验性)
- 支持在线扩容与缩容, 支持数据去重

注意事项

- CoW 机制带来额外 CPU 与内存开销
- RAID 5/6 在部分内核版本仍不稳定, 生产环境慎用
- 碎片管理、平衡 (balance) 等操作需要定期维护
- 特性丰富但复杂度高, 学习曲线较陡

六、ZFS 与其他常用文件系统

ZFS

ZFS 最初由 Sun 为 Solaris 开发, 现以 OpenZFS 形式在 Linux 上广泛使用 (但因其 CDDL 许可协议未进入 Linux 主线内核, 需单独安装)。它是存储领域"天花板"级别的文件系统:

- 数据自愈: 端到端校验和, 静默损坏自动修复
- 快照与克隆、内置 RAID (RAID-Z)、压缩、去重
- 存储池 (zpool) 统一管理, 弹性扩展
- 缺点: 内存占用高 (建议 1GB 内存/1TB 存储)、许可协议问题、不能原生开机引导 (通常需要单独 /boot 分区)

其他常见文件系统

文件系统	特点	典型用途
FAT32 / vfat	跨平台兼容性极佳；单文件最大 4GB；无权限、无日志	U 盘、SD 卡、EFI 系统分区
NTFS	Windows 原生；支持大文件、权限、加密压缩	双系统共享、访问 Windows 盘
exFAT	无 4GB 限制，兼容性好；无日志	大容量移动硬盘、相机
f2fs	为闪存（Flash）优化的文件系统	SSD、eMMC、嵌入式设备
ISO 9660	光盘文件系统	光驱、ISO 镜像
NFS / CIFS	网络文件系统，远程共享	NAS、集群共享存储
OverlayFS	联合挂载文件系统	Docker 默认存储驱动

七、特殊文件系统：swap / tmpfs / 虚拟文件系统

swap 交换分区

swap 不是传统意义上的文件系统，而是用于扩展物理内存的"交换空间"，充当虚拟内存。当物理内存不足时，内核把不活跃的内存页换出到 swap。

- 可作为独立分区，也可作为文件（swapfile）
- 过度使用 swap 会导致性能急剧下降（磁盘比内存慢几个数量级）
- 挂载在 /proc/swaps，可用 free -h 查看使用情况

tmpfs

tmpfs 是基于内存（RAM）的临时文件系统，读写速度极快，但重启后数据全部丢失，且占用内存容量。

- /tmp、/run、/dev/shm 等目录常用 tmpfs
- 容器和某些缓存场景大量使用 tmpfs 提升性能

虚拟文件系统

文件系统	挂载点	作用
proc	/proc	进程与内核信息，如 /proc/meminfo、/proc/cpuinfo
sysfs	/sys	内核与设备信息，设备树视图
devtmpfs	/dev	设备文件（/dev/sda、/dev/tty 等）
cgroup	/sys/fs/cgroup	资源隔离与控制（容器技术基石）

这些虚拟文件系统并不占用真实磁盘，而是内核对外暴露的接口，体现了 Linux"一切皆文件"的设计哲学。

八、核心概念：inode 与日志

inode（索引节点）

inode 是文件系统中核心的数据结构，每个文件（包括目录）都对应一个 inode。它记录文件的元数据：权限、属主、时间戳、大小、链接计数，以及指向数据块的指针。

关键理解：文件名并不是文件本身。文件名只是目录中的一条"目录项"（dentry），它指向 inode；inode 再指向真正数据块。

- ls -li 查看 inode 编号
- df -li 查看 inode 使用量——当 inode 耗尽时，即使磁盘还有空间，也会报"No space left on device"
- 每个文件系统在创建时就分配了固定数量的 inode，大量小文件容易耗尽 inode

日志（Journaling）

日志功能是现代文件系统（ext3/ext4、XFS、Btrfs）的标配：在真正写数据之前，先把"将要做的修改"记录到日志区，然后才执行写入。断电或崩溃时，系统通过回放日志快速恢复到一致状态，避免了漫长的全盘检查。

- ext4/XFS: 传统日志 (metadata journaling)
- Btrfs/ZFS: 基于 CoW 的日志化机制，无需独立日志区

九、主流文件系统对比与选型

核心对比表

特性	ext4	XFS	Btrfs	ZFS
最大文件	16 TB	8 EB	16 EB	16 EB
最大分区	1 EB	8 EB	16 EB	256 ZB (理论)
日志	有	有	CoW 机制	CoW 机制
快照	无	无	有	有
透明压缩	无	无	有	有
数据校验	无	有 (元数据)	有 (数据+元数据)	有 (端到端)
内置 RAID	无	无	有 (5/6 实验性)	有 (RAID-Z)
在线扩容	可扩可缩	只能扩	可扩可缩	可扩可缩
默认发行版	Debian/Ubuntu	RHEL/CentOS	openSUSE/Fedora	FreeBSD/Solaris
稳定性	非常成熟	非常成熟	核心功能成熟	非常成熟
资源占用	低	中	中高	高

选型建议

场景	推荐	理由
桌面 / 通用服务器	ext4	成熟稳定，兼容性最好
系统盘 / 启动分区	ext4 或 XFS	GRUB 兼容性最好，可靠性优先
大型数据库	XFS	高并发 I/O 性能优异
视频 / 媒体 / 大文件	XFS	大文件处理性能最佳
需要快照 / 压缩 / 回滚	Btrfs	高级功能最丰富
NAS / 数据完整性要求极高	ZFS 或 Btrfs	自愈、校验、去重
云主机根分区	ext4	通用性与工具链最成熟

常用命令

```
# 创建文件系统
```

```
sudo mkfs.ext4 /dev/sda1
sudo mkfs.xfs /dev/sda1
sudo mkfs.btrfs /dev/sda1
sudo mkfs.fat -F32 /dev/sda1      # FAT32 (EFI 分区)
sudo mkswap /dev/sda2 && sudo swapon /dev/sda2  # swap

# 挂载与查看
sudo mount /dev/sda1 /data
df -Th          # 查看挂载情况与文件系统类型
df -i           # 查看 inode 使用量
blkid           # 查看 UUID 与文件系统类型
cat /proc/filesystems # 查看内核支持的文件系统

# 扩容
sudo resize2fs /dev/sda2      # ext4 扩容 (作用于设备)
sudo xfs_growfs /data         # XFS 扩容 (作用于挂载点)
```

第三部分 Linux 文件类型

一、核心哲学：一切皆文件

Linux 沿袭 Unix 的设计哲学："一切皆文件" (Everything is a file)。不仅普通数据是文件，目录、设备、进程信息、通信管道，统统以文件的形式暴露给用户和应用程序。这种统一抽象让系统变得简单而强大：同样的 open/read/write 系统调用，既可以操作文本文件，也可以操作磁盘、终端甚至网络连接。

ls -l 输出结果的第一位字符，就是用来标识文件类型的。

二、七种文件类型详解

1. 普通文件 (-)

最常见的文件类型，存储文本、图片、数据库、二进制程序等任何字节流。创建方式：touch、vim、echo > 等。

2. 目录 (d)

目录本质上也是一种特殊文件，其内容是"文件名 → inode"的映射表。目录可以包含其他文件和子目录，形成树状结构。创建方式：mkdir。

- 目录的读权限控制能否列出内容
- 目录的执行权限控制能否进入 (cd)

3. 符号链接 (l)

符号链接（软链接）是一个小文件，其内容是"指向另一个文件的路径"。相当于 Windows 的快捷方式。创建方式：ln -s。

- 可以跨文件系统、可以链接目录
- 目标被删除后，链接变成"悬空链接" (dangling)，访问报错
- `ls -l` 显示为 `lrwxrwxrwx`，并带 `→` 指向目标

4. 块设备 (b)

以固定大小数据块为单位进行带缓冲、随机访问读写的设备文件，主要代表存储设备。位于 `/dev` 目录。

- `/dev/sda` (硬盘)、`/dev/nvme0n1` (NVMe SSD)、`/dev/sr0` (光驱)
- 块设备是分区的载体，分区表就写在块设备上

5. 字符设备 (c)

以字节流为单位、无缓冲地逐个字符传输数据的设备文件。

- `/dev/tty` (终端)、`/dev/null` (黑洞)、`/dev/urandom` (随机数)
- `/dev/null` 常用来丢弃输出：`command > /dev/null 2>&1`
- 判断原则：一个设备如果不是块设备，就是字符设备

6. 套接字 (s)

套接字文件用于进程间通信 (IPC)，支持双向通信，是网络和本地通信的基础。

- `/var/run/docker.sock`: Docker 守护进程的本地通信入口
- 数据库 (如 MySQL/PostgreSQL) 常用 Unix socket 提供本地连接

7. 命名管道 (p)

命名管道 (FIFO) 用于进程间单向、先入先出的数据传递，无需经过网络。创建方式：`mkfifo`。

- 一端写入，另一端读取
- 常用于 shell 脚本中进程间传数据，轻量高效

三、如何识别文件类型

方法一：ls -l 首字符

```
$ ls -l /dev/sda /etc/passwd /bin/ls /var/run/docker.sock
brw-rw---- 1 root disk 8, 0 ... /dev/sda      ← b 块设备
-rw-r--r-- 1 root root   ... /etc/passwd   ← - 普通文件
lrwxrwxrwx 1 root root   ... /bin/ls -> /usr/bin/ls ← l 软链接
srw-rw---- 1 root docker ... /var/run/docker.sock ← s 套接字
```

方法二：file 命令（按内容识别）

file 命令通过读取文件的"魔数"（magic number，文件头部的特定字节序列）来识别真实类型，不依赖扩展名——Linux 不靠扩展名判断文件类型。

```
$ file /etc/passwd
/etc/passwd: ASCII text
$ file /bin/ls
/bin/ls: ELF 64-bit LSB pie executable, x86-64, ...
$ file image.jpg
image.jpg: JPEG image data, JFIF standard 1.01, ...
```

方法三：stat 命令

stat 可查看文件的完整元数据，第一行直接给出文件类型。

```
$ stat /etc/passwd
File: /etc/passwd
Size: 2912      Blocks: 8      IO Block: 4096   regular file
...
```

常用查找命令

```
find / -type f      # 查找普通文件
find / -type d      # 查找目录
find / -type l      # 查找软链接
find / -type s      # 查找套接字
find / -type b / -type c / -type p  # 设备与管道
file /dev/*         # 查看设备类型
```

四、硬链接与软链接

链接是理解 Linux 文件系统的关键。两种链接方式表面上都能让一个文件出现在多个位置，但原理截然不同。

硬链接

硬链接是为同一 inode 增加一个新的目录项。创建硬链接后，两个名字共享同一个 inode、同一份数据，地位完全平等，没有"原件"与"链接"之分。

- 创建：ln 源文件 链接名（不加 -s）
- ls -li 查看，两个名字 inode 相同，链接计数（硬链接数）增加
- 删除其中一个名字，数据仍保留（只要链接计数不为 0）
- 不能跨文件系统创建（inode 只在同一文件系统内有效）
- 不能链接目录（会造成循环）

软链接（符号链接）

软链接是一个独立的小文件（拥有自己的 inode），其内容只是一个目标路径。访问软链接时，内核会顺着路径找到目标文件。

- 创建：ln -s 目标路径 链接名

- 可以跨文件系统、可以链接目录
- 目标被删除或移动后，软链接变成悬空链接（broken）
- ls -l 显示 l 开头和 → 指向
- 版本管理中常用：/opt/myapp/current → releases/v1.2，切换版本只需改链接

硬链接 vs 软链接对比

对比项	硬链接	软链接（符号链接）
inode	与原文件共享同一 inode	拥有独立 inode
本质	同一文件的多重名字	存储目标路径的小文件
跨文件系统	不支持	支持
链接目录	不允许	允许
目标被删除	数据仍在（计数减 1）	链接失效（悬空）
文件大小	与原文件相同	等于路径字符串长度
识别方式	与普通文件无区别	ls -l 显示 l 和 →
典型用途	备份去重（rsnapshot）	快捷方式、版本切换

```
# 实践验证
echo "hello" > original.txt
ln original.txt hard.txt          # 创建硬链接
ln -s original.txt soft.txt      # 创建软链接

ls -li original.txt hard.txt soft.txt
# 1234567 -rw-r--r-- 2 ... original.txt  ← inode 相同, 计数 2
# 1234567 -rw-r--r-- 2 ... hard.txt
# 1234568 lrwxrwxrwx 1 ... soft.txt -> original.txt  ← 独立 inode

rm original.txt
cat hard.txt      # 正常输出: hello (数据仍存在)
cat soft.txt      # 报错: No such file or directory (悬空链接)
```

第四部分 总结：三者区别一目了然

一、三个概念的分层总结

维度	分区格式（分区表）	文件系统	文件类型
回答的问题	盘怎么切	切出来的空间怎么存	存的都是什么
所在层次	磁盘最底层	分区之上	文件系统之内
常见类型	MBR、GPT	ext4、XFS、Btrfs、ZFS、swap、vfat...	普通文件、目录、软链接、块/字符设备、套接字、管道
管理工具	fdisk、gdisk、parted	mkfs.*、mount、resize2fs、xfs_growfs	ls -l、file、stat、find -type
修改影响	重新分区会破坏数据	换文件系统需格式化（破坏数据）	只是文件本身的属性
粒度	整块磁盘（几个分区）	单个分区/逻辑卷	单个文件

三者的典型操作链路：

1. 分区：sudo parted /dev/sdb mklabel gpt	← 决定分区格式
2. 分区：sudo parted /dev/sdb mkpart primary ext4 1MiB 100%	← 切出分区
3. 格式化：sudo mkfs.ext4 /dev/sdb1	← 写入文件系统
4. 挂载：sudo mount /dev/sdb1 /data	← 挂载到目录树
5. 使用：echo hello > /data/a.txt	← 在文件系统中创建文件
6. 识别：ls -l /data/a.txt	（第一个字符 - 即文件类型）

二、常见误区辨析

误区	正解
"MBR 和 ext4 都是分区格式"	MBR/GPT 是分区表格式；ext4/XFS 是文件系统。分区表在磁盘层面，文件系统在分区层面
"Linux 靠扩展名识别文件类型"	Linux 靠文件内容（魔数）和 inode 元数据识别，扩展名只是给用户看的约定
"格式化了分区就丢失分区表"	格式化（mkfs）写的是文件系统；分区表是 fdisk/gdisk/parted 管理的，两者是不同的写入区域
"硬盘还有空间却报磁盘满"	可能是 inode 耗尽（df -i 查看），大量小文件吃光

	inode 所致
"swap 是文件系统"	swap 是交换空间，没有文件系统的目录结构，不能挂载存储文件
"硬链接就是复制一份文件"	硬链接不复制数据，只是同一 inode 的另一个名字；修改任何一边，两边数据同步变化

三、实操速查

查看命令清单

```
lsblk -f          # 一次看全：磁盘 → 分区 → 文件系统类型 → 挂载点
lsblk -o NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT  # 自定义列
fdisk -l          # 分区表详情 (Disk label type: gpt/dos)
blkid             # 各分区 UUID 与文件系统类型
df -Th           # 已挂载文件系统的使用情况
df -i            # inode 使用情况
cat /proc/filesystems  # 内核支持的文件系统
ls -l /dev        # 查看设备文件类型 (b/c)
file /path        # 按内容识别文件真实类型
stat /path        # 查看 inode 与文件类型详情
```

选型一句话

新磁盘用 GPT 分区；通用场景用 ext4；大文件、数据库、红帽系服务器用 XFS；要快照、压缩、回滚用 Btrfs；数据完整性极致要求用 ZFS。