存储管理

一、基础知识

1、介质

- 硬盘有机械硬盘(HDD)和固态硬盘(SSD)之分。

2、机械硬盘（HDD）

- 机械硬盘即是传统普通硬盘，主要由：磁头组件、磁头驱动机构、盘片组、控制电路和接口等几个部分组成。
- 机械硬盘中所有的盘片都装在一个旋转轴上，每张盘片之间是平行的，在每个盘片的存储面上有一个磁头，磁头与盘片之间的距离比头发丝的直径还小，所有的磁头联在一个磁头控制器上，由磁头控制器负责各个磁头的运动。
- 盘片的数量和每个盘片的存储容量确定了磁盘的总容量。
- sas start

- 磁头：负责读写数据
- 盘片：保存写入的数据
- 主轴：转动盘片，将盘片上的指定位置移动到磁头下
- 控制电路：控制硬盘的速度，磁头臂的移动、向磁头下发命令等

3、固态硬盘（SSD）

- 固态驱动器（Solid State Disk或Solid State Drive，简称SSD），俗称固态硬盘，固态硬盘是用固态电子存储芯片阵列而制成的硬盘。
- 固态硬盘不像传统的硬盘采用磁性材料存储数据，而是使用flash技术存储信息。其特点就是断电后数据不消失。
- 固态硬盘没有内部机械部件，并不代表其生命周期无限，flash闪存是非易失存储器，可以对称为块的存储器单元块进行擦写和再编程。任何flash器件的写入操作只能在空或已擦除的单元内进行，所

以大多数情况下，在进行写入操作之前必须先执行擦除。因擦除次数有限，所以固态硬盘也是有生命周期的。

SSD性能优势

- 响应时间短：SSD硬盘内部没有机械运动部件，省去了寻道时间和机械延迟时间，可更快捷的响应读写请求。
- 读写速率高：SSD通过内部控制器计算出数据的存放位置，并进行速写操作，省去了机械操作时间，大大提高了读写效率。
- 不会产生噪音

4、硬盘关键指标

- 硬盘容量(volume)：容量的单位为兆字节或千兆字节。影响硬盘容量的因素是盘片数量和单盘容量。
- 转速(Rotational speed)：硬盘的转速是硬盘盘片每分钟转过的圈数，单位为RPM(rotation per minute)，一般硬盘的转速达到5400RPM/7200RPM。SCSI接口硬盘转速可达10000~15000RPM。
- 平均访问时间(Average Access Time)=平均寻道时间+平均等待时间。
- 数据传输率(Data Transfer Rate)：硬盘的数据传输率是指硬盘读写数据的速度。主要有内部传输率和外部传输率。
- IOPS(Input/Output Per Second)：即每秒输入输出量。

5、磁盘接口类型

1. SATA（Serial ATA）

- **特点**：
  - 最常见的家用接口，价格便宜，速度中等（SATA 3.0最高6Gbps）。
  - 兼容性强，支持机械硬盘（HDD）和固态硬盘（SSD）。
- **适用场景**：
  - **普通家用电脑**：办公、上网、看视频。

    - **轻度游戏**：预算有限时，用SATA SSD装系统+游戏。
    - **外接硬盘盒**：移动硬盘常用SATA转USB。
- **举个栗子🌰**：
  你的旧笔记本升级，加一块500GB的SATA固态硬盘，开机速度从1分钟变10秒。

---

2. SAS（Serial Attached SCSI）

- **特点**：
    - 企业级接口，速度快（SAS 4.0可达22.5Gbps），稳定性高。
    - 支持多设备串联（一条线接多个硬盘），适合7×24小时高强度运行。
- **适用场景**：
    - **企业服务器**：数据库、虚拟化、云计算存储。
    - **数据中心**：需要高并发读写（比如淘宝双十一的订单处理）。
    - **关键业务存储**：银行、医院等不能宕机的场景。
- **举个栗子🌰**：
  阿里云的服务器硬盘柜里，插满SAS接口的硬盘，保证每秒处理几十万次请求。

---

3. PCIe/NVMe（M.2形态常见）

- **特点**：
    - 直接通过PCIe通道连接CPU，速度爆炸（NVMe协议下可达3500MB/s以上）。
    - 体积小（像口香糖），无需数据线，直接插主板。
- **适用场景**：
    - **高性能电脑**：电竞主机、视频剪辑工作站。
    - **大型游戏加载**：用NVMe SSD秒开《赛博朋克2077》。
    - **专业创作**：4K/8K视频剪辑、3D渲染缓存盘。
- **举个栗子🌰**：
  你买了一块1TB的M.2 NVMe SSD，装进PS5里，游戏加载时间从30秒缩到3秒。

---

5. IDE（已淘汰）

- **特点**：
    - 老式并行接口，速度慢（最高133MB/s），线又宽又丑。
    - 已被SATA全面取代。
- **历史场景**：
    - **2000年前的电脑**：Windows 98/XP时代的机械硬盘。
    - **古董设备**：老式工控机、ATM机（现在基本绝迹）。

---

6. SCSI（老式，非SAS）

- **特点**：
    - 早期企业级接口，并行传输，速度慢（最高320MB/s），需要专用卡。
    - 已被SAS淘汰。
- **历史场景**：
    - **90年代服务器**：小型机、Unix工作站。
    - **老式磁带机/光驱**：博物馆级设备。

总结：不同接口的适用人群

接口类型	适合谁?	典型设备	价格
SATA	普通用户、预算有限	家用电脑、外置硬盘	便宜
SAS	企业IT部门、数据中心	服务器、存储阵列	很贵
PCIe/NVMe	游戏玩家、专业创作者	电竞PC、MacBook Pro	中等~贵
U.2	土豪企业、云服务商	全闪存存储服务器	非常贵
IDE/SCSI	怀旧爱好者、古董设备维护	老式电脑、工控机	已淘汰

如何选择?

- **家用/办公**：SATA固态或机械硬盘。
- **游戏/设计**：NVMe SSD（M.2）。
- **企业服务器**：SAS或U.2 SSD。
- **钱多任性**：全上NVMe+U.2，享受飞一般的感觉 💸。

希望帮你理清思路！如果有具体需求，可以再聊～ 😊

6、设备的命名规则

一、核心规则总结

Linux 中硬件设备（如硬盘、U盘）会被系统识别为文件，存放在 `/dev/` 目录下。
命名格式：`/dev/[前缀][字母][数字]`

- **前缀**：表示设备类型（比如 `sd` 是 SATA/SCSI 硬盘，`hd` 是 IDE 硬盘）。
- **字母**：表示第几块硬盘（`a`=第一块，`b`=第二块，依此类推）。
- **数字**：表示分区编号（主分区/扩展分区用 1-4，逻辑分区从 5 开始）。

二、具体规则拆解

1. 前缀：设备类型

前缀	设备类型	举例
`sd`	SATA、SCSI、USB、SAS 接口的硬盘	`/dev/sda`（第一块SATA盘）
`hd`	IDE 接口硬盘（老式，已淘汰）	`/dev/hda`（第一块IDE盘）
`vd`	虚拟化硬盘（如 KVM 虚拟机）	`/dev/vda`（虚拟机第一块盘）
`nvme`	NVMe 协议的高速固态硬盘	`/dev/nvme0n1`（第一块NVMe盘）

2. 字母：硬盘顺序

- 系统按检测到的顺序分配字母：

  第一块硬盘 → `a`，第二块 → `b`，依此类推。

  - 比如插两块 SATA 硬盘，系统识别为 `/dev/sda` 和 `/dev/sdb`。

3. 数字：分区编号

- **主分区/扩展分区**：只能用 `1-4`（一块硬盘最多4个）。
- **逻辑分区**：从 `5` 开始编号（必须创建在扩展分区内）。

举个实际例子🌰：
假设一块 SATA 硬盘（`/dev/sda`）的分区如下：

- `sda1`：主分区（比如装 Windows 系统）。
- `sda2`：扩展分区（相当于一个容器）。
- `sda5`、`sda6`：扩展分区内的逻辑分区（比如存数据）。

三、常见问题解答

1. 为什么逻辑分区从 5 开始?

- **历史原因**：早期的硬盘分区表设计限制，主分区最多4个，逻辑分区为了不和主分区冲突，从5开始编号。
- **扩展分区是虚拟容器**：它本身不存数据，只是用来装逻辑分区的“盒子”。

2. 如果一块硬盘只有 3 个主分区，逻辑分区编号会怎样?

- 假设硬盘分区为：`sda1`（主）、`sda2`（主）、`sda3`（扩展），那么：
  - 扩展分区内的逻辑分区会从 `sda5` 开始（即使 `sda4` 没用）。
  - 因为主分区和扩展分区共用 1-4 的编号。

3. 多硬盘的情况如何命名?

- 比如插入三块 SATA 硬盘：
  - 第一块 → `/dev/sda`
  - 第二块 → `/dev/sdb`
  - 第三块 → `/dev/sdc`
- 每块硬盘的分区独立编号，比如第二块硬盘的第一个主分区是 `/dev/sdb1`。

4. U盘、移动硬盘如何命名?

- 和 SATA 硬盘规则相同，比如插入的U盘可能是 `/dev/sdd`，它的第一个分区是 `/dev/sdd1`。

四、命名规则速查表

设备名称	含义
`/dev/sda`	第一块 SATA/SCSI 硬盘（整块硬盘）
`/dev/sda1`	第一块 SATA 硬盘的第一个主分区
`/dev/sda5`	第一块 SATA 硬盘的第一个逻辑分区
`/dev/vdb`	虚拟机的第二块虚拟硬盘
`/dev/nvme0n1`	第一块 NVMe 硬盘的第一个分区

五、实际场景演练

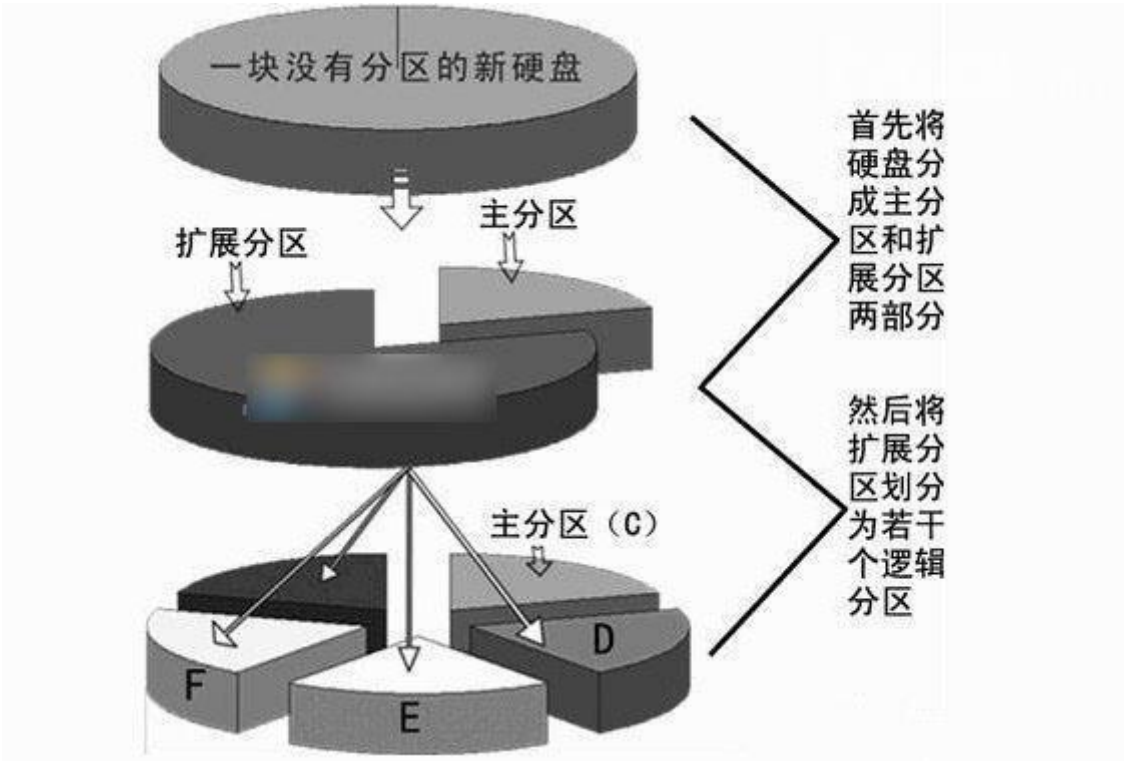
假设你的电脑配置如下：

- 一块 SATA 固态硬盘（装系统）
- 一块 SATA 机械硬盘（存数据）
- 一个插入的 U 盘

设备命名结果：

1. 固态硬盘 → `/dev/sda`
    - 分区：`sda1`（主分区，装 Linux 系统）
2. 机械硬盘 → `/dev/sdb`
    - 分区：`sdb1`（主分区）、`sdb2`（扩展分区）、`sdb5`（逻辑分区）
3. U盘 → `/dev/sdc`
    - 分区：`sdc1`（U盘唯一分区）

7、磁盘分区类型



理解主分区、扩展分区和逻辑分区的概念

- # 一、主分区（Primary Partition）

1. 定义

- **直接可用的分区**，像独立的房间，能直接存数据或装系统。
- **数量限制**：一块硬盘最多只能有 **4个主分区**（受早期分区表设计限制）。

2. 实际例子

假设你有一块新硬盘1t，想分3个区：

1. 系统盘（装Windows） → 主分区1（C盘） 200
2. 软件盘（装游戏） → 主分区2（D盘）100
3. 备份盘 → 主分区3（E盘）100

4. 空着不用 → 主分区4（如果以后需要）

此时硬盘分区表：

```
主分区1（C盘）
主分区2（D盘）
主分区3（E盘）
未分配（剩余空间）
```

二、扩展分区（Extended Partition）

1. 定义

- **虚拟容器**：为了解决"主分区最多4个"的限制，将其中一个主分区变成扩展分区。
- 特点：
  - 扩展分区本身不能存数据，必须再划分成 **逻辑分区。**
  - 一个硬盘只能有 **1个扩展分区。**

2. 实际例子

假设你想在一块硬盘上分5个区：

1. 主分区1（C盘，装系统）
2. 扩展分区（占剩余空间）
   - 扩展分区内划分：
     - 逻辑分区1（D盘，装游戏）
     - 逻辑分区2（E盘，存电影）
     - 逻辑分区3（F盘，备份）

此时硬盘分区表：

```
主分区1（C盘）
扩展分区（包含D、E、F盘）
```

三、逻辑分区（Logical Partition）

1. 定义

- **扩展分区内的子分区**，像大仓库里隔出的小房间。
- **数量无限制**（理论上可无限分，但系统会限制，比如Windows最多128个）。
- **编号从5开始**（主分区用1-4，逻辑分区从5开始）。

2. 实际例子

假设一块硬盘的分区如下：

```
主分区1（C盘，系统）
主分区2（D盘，软件）
扩展分区（占剩余空间）
    → 逻辑分区5（E盘，电影）
    → 逻辑分区6（F盘，备份）
```

为什么编号从5开始?

- 主分区和扩展分区占用1-4的编号，逻辑分区从5开始。

四、分区方案对比

分区类型	作用	数量限制	能否直接存数据	编号规则
主分区	直接装系统或存数据	最多4个	✅ 能	1-4
扩展分区	容器，内部再分逻辑分区	只能1个	❌ 不能	占用1-4中的一个
逻辑分区	扩展分区内的实际可用分区	无限制（理论上）	✅ 能	从5开始

五、实际场景演练

场景1：普通家用电脑分区（单系统）

- **主分区1**：C盘（装Windows系统，100GB）
- 扩展分区：剩余空间
  - **逻辑分区5**：D盘（500GB，装游戏）
  - **逻辑分区6**：E盘（1TB，存电影）

场景2：双系统（Windows + Linux）

- **主分区1**：C盘（Windows系统，100GB）
- **主分区2**：/dev/sda2（Linux系统，200GB）
- 扩展分区：剩余空间
  - **逻辑分区5**：D盘（数据盘）
  - **逻辑分区6**：/home（Linux用户数据盘）

六、常见问题解答

1. 为什么要有扩展分区?

- **解决主分区数量限制**：比如你想分5个区，必须用扩展分区+逻辑分区。

2. 扩展分区能直接存数据吗?

- **不能**！扩展分区只是一个“盒子”，必须划分成逻辑分区才能用。

3. 逻辑分区能装系统吗?

- **可以**：但有些系统（如老版本Windows）必须装在主分区。

4. 如何查看自己的分区类型?

- **Windows**：右键“此电脑” → 管理 → 磁盘管理。
- **Linux**：终端输入 `lsblk` 或 `fdisk -l`。

总结比喻：

- **主分区**：独立房间（直接住人）。
- **扩展分区**：一个大仓库（不能住人，但可以隔成小房间）。
- **逻辑分区**：仓库里的小房间（实际使用）。

一块硬盘最多只能有4个“独立房间”（主分区），但你可以把一个房间改造成“仓库”（扩展分区），再在里面无限分隔小房间（逻辑分区）。

```
[root@xnha CentOS-8-1-1911-x86_64-dvd]# fdisk -l
Disk /dev/nvme0n1：20 GiB，21474836480 字节，41943040 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x69a55b75

#1. 整块硬盘：一栋房子（/dev/nvme0n1）•
•房子大小：20GB（总空间）。
•户型设计：老式户型（磁盘标签类型：dos，即MBR分区表）。

设备             启动     起点      末尾      扇区 大小 Id 类型
/dev/nvme0n1p1 *        2048   2099199   2097152    1G 83 Linux
/dev/nvme0n1p2       2099200 41943039 39843840   19G 8e Linux LVM

#2. 分区结构：房子里划分房间
•房间1：主分区（/dev/nvme0n1p1）•
•用途：独立房间，直接使用。
•大小：1GB（装Linux系统启动文件）。
•类型：普通房间（类型83 Linux）。
•位置：房子的前1GB空间（起点2048扇区）。
比喻：进门左手边的独立卧室（放重要物品——系统启动文件）。

•房间2：主分区（/dev/nvme0n1p2）•
•用途：改造成了「大仓库」（类型 8e Linux LVM），内部可以灵活分隔。
•大小：19GB（占房子剩余空间）。
•内部结构：

Disk /dev/mapper/cs_locahost-root：17 GiB，18249416704 字节，35643392 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
•仓库里的小房间1：/dev/mapper/cs_locahost-root（17GB，根分区，装系统和软件）

Disk /dev/mapper/cs_locahost-swap：2 GiB，2147483648 字节，4194304 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
•仓库里的小房间2：/dev/mapper/cs_locahost-swap（2GB，虚拟内存分区，类似Windows的页面文件）。
```

比喻：右手边的大仓库被改造成两个功能区：

•客厅+厨房（root）•：17GB，日常活动的主要区域（系统运行的核心）。
•应急储物间（swap）•：2GB，临时存放用不到的东西（内存不足时缓存数据）。
#3. 为什么没有扩展分区？
•用了LVM（逻辑卷管理）•：直接把主分区2变成「弹性仓库」，无需传统的扩展分区。
•优势：LVM可以动态调整分区大小（传统分区不能），比如未来给客厅扩容时，无需搬家具（不丢数据）。
#4. 总结结构图
整栋房子（20GB）
├── [主分区1] 独立卧室（1GB，启动文件） → /dev/nvme0n1p1
└── [主分区2] 大仓库（19GB，LVM） → /dev/nvme0n1p2
    ├── 客厅+厨房（17GB，系统核心） → /dev/mapper/cs_locahost-root
    └── 应急储物间（2GB，虚拟内存） → /dev/mapper/cs_locahost-swap
#关键点
•LVM 比传统分区更灵活：就像把仓库改造成可伸缩的集装箱，能随时调整大小、合并或拆分空间。
•为什么用MBR（dos）•：虽然硬盘是NVMe（新接口），但分区表用了老式的MBR（可能为了兼容性）。
•没有逻辑分区（sda5这种）•：因为LVM替代了传统的扩展分区方案，直接在主分区上玩“空间魔法”。

8、扇区又是什么

扇区（Sector）的通俗解释

1. 定义

- **扇区是硬盘存储的最小单位**，就像书本的“一页纸”，数据必须按“页”读写。
- **大小固定**：传统硬盘每扇区 **512字节**，现代大容量硬盘可能用 **4KB（4096字节）**。

---

2. 比喻理解

把硬盘想象成一本书：

- **扇区** → 书的一页（最小阅读单位，不能只读半页）。
- **磁道** → 书的一行字（硬盘盘片上的同心圆轨道）。
- **柱面** → 多本书叠在一起（多个盘片的同一磁道组成柱面）。

举个栗子🌰：
如果你要存一首5MB的MP3歌曲，硬盘会把它拆成很多“页”（扇区），每页存512字节，总共需要约10,000个扇区（5MB ÷ 512B ≈ 9766，实际会更多）。

---

3. 为什么扇区重要?

- **读写的最小单位**：即使你只修改1字节，硬盘也要读写整个扇区。
- **效率问题**：扇区太小 → 管理开销大；扇区太大 → 浪费空间（比如存1字节却占用4KB）。
- **坏道（Bad Sector）**：如果某扇区损坏，这块区域的数据会丢失（就像书页被撕掉）。

---

4. 实际例子

查看你提供的 `fdisk -l` 输出：

- 硬盘 `/dev/nvme0n1` 总共有41,943,040个扇区，每个扇区512字节。
  - 总容量 = 41,943,040 × 512字节 ≈ 20GB。
- 分区 `/dev/nvme0n1p1` 的起点是 2048扇区，末尾是 2,099,199扇区。

- 分区大小 = (2,099,199 - 2048 + 1) × 512字节 ≈ 1GB。

---

5. 扇区大小演变

- **传统扇区**：512字节（几十年标准，兼容性好）。
- 高级格式（Advanced Format）：4KB（现代大硬盘用，提升效率）。
  - **注意**：4KB扇区硬盘在旧系统上可能有兼容问题！

总结

- **扇区是硬盘的"最小存储单元"**，数据按扇区读写。
- **就像书页**：读数据必须翻到某一页，写数据必须整页修改。
- **实际影响**：文件系统、分区工具、硬盘性能都和扇区设计息息相关。

9、磁盘分区表MBR 和GPT的区别

- MBR的意思就是主引导记录，目前我们使用的硬盘绝大部分都是512字节的一个扇区，MBR分区表中逻辑地址以32位二进制表示，所以最大只能表示232（2的32次方）个地址，最大单分区容量为232*512（2的32次方乘于512）字节=2048GB，也就是我们通常说的MBR单分区最大支持2TB
- GPT的意思是GUID Partition Table，即"全局唯一标识磁盘分区表"，是源自EFI标准的一种较新的磁盘分区表结构的标准，GPT提供了更加灵活的磁盘分区机制。支持大于2TB单分区、多分区、分区表自带备份。而GPT是另一种更先进的磁盘系统分区方式，它的出现弥补了MBR这个缺点，最大支持18EB的硬盘，是基于UEFI使用的磁盘分区架构

1、MBR格式分区

- 主分区数量不能超过4个；
- 分区大小无法超过2TB容量；
- 支持安装所有的Windows操作系统；

2、GPT格式分区

- 磁盘分区数量几乎无限制，但是Windows系统只允许最多128个分区；
- 支持2TB以上容量的硬盘；
- 仅支持安装64位操作系统，因为UEFI引导启动只支持64位操作系统，如果想要用EFI引导32位系统，貌似必须主板开启CSM兼容模块支持；
- GPT格式分区表自带备份；

3、mbr和gpt分区最大支持多大容量

- MBR分区表的硬盘最大只能划分为4个主分区，当然逻辑分区是可以支持分到四个以上的，并且最大支持2TB的硬盘，如果需要分区的硬盘容量超过了2TB容量，例如3TB、4TB、6TB、8TB、10TB等以上容量，则需要使用GPT分区表类型，不会受到硬盘容量大小、分区数量的限制，不过在Windows系统上由于系统的限制，支持最多128个GPT磁盘分区，并且gpt格式是没有主分区和逻辑分区这个概念的。

4. MBR（老式目录）

- **比喻**：一本老书的目录，写在第一页（硬盘的第一个扇区）。
- 特点：
  - **目录长度有限**：最多记录4个章节（主分区）。

- **扩展目录**：如果章节超过4个，可以指定其中一个章节为“扩展目录”（扩展分区），里面再分小节（逻辑分区）。
- **容量限制**：书的总大小不能超过2TB（相当于只能写2万亿字）。
- **单点故障**：如果第一页（目录页）被撕毁，整本书就废了（数据丢失）。

举个栗子🌰：
一本老书《MBR使用指南》目录：

```
1. 主分区1（第1-100页：装系统）
2. 主分区2（第101-200页：存文件）
3. 扩展分区（第201-500页）
   → 逻辑分区5（第201-300页：电影）
   → 逻辑分区6（第301-500页：游戏）
```

---

5. GPT（新式目录）

- **比喻**：一本智能电子书的目录，写在开头和结尾（分区表有备份）。
- 特点：
  - **目录无限长**：支持128个以上章节（分区），无需扩展分区。
  - **超大容量**：书的总大小最高18EB（1EB=10亿GB，相当于能写18亿亿字）。
  - **双保险**：目录在开头和结尾各存一份，一份损坏了还能用另一份恢复。
  - **兼容性**：必须用新式阅读器（UEFI主板）才能读这本书。

举个栗子🌰：
一本新书《GPT使用指南》目录：

```
1. 分区1（第1-1000页：系统）
2. 分区2（第1001-2000页：照片）
3. 分区3（第2001-3000页：视频）
...
128. 分区128（第...页：备份）
```

---

6. 核心区别对比表

对比项	MBR（老书目录）	GPT（新书目录）
最大分区数	4主分区（或3主+1扩展）	理论上无限（Windows限128）
最大容量	2TB	18EB（天文数字）
目录备份	无（目录丢失=全书报废）	有（开头和结尾各存一份）
兼容性	支持老设备（BIOS主板）	需要新设备（UEFI主板）
适用场景	老旧电脑、小硬盘	新电脑、大硬盘、服务器

二、管理磁盘

Linux磁盘管理实验手册：添加磁盘、分区、格式化与挂载

实验目标

1. 掌握在VMware虚拟机中添加新磁盘的方法。
2. 学习使用`fdisk`工具对磁盘进行分区。
3. 掌握创建文件系统（格式化）和挂载磁盘的操作。
4. 理解MBR分区表的限制及扩展分区的使用。

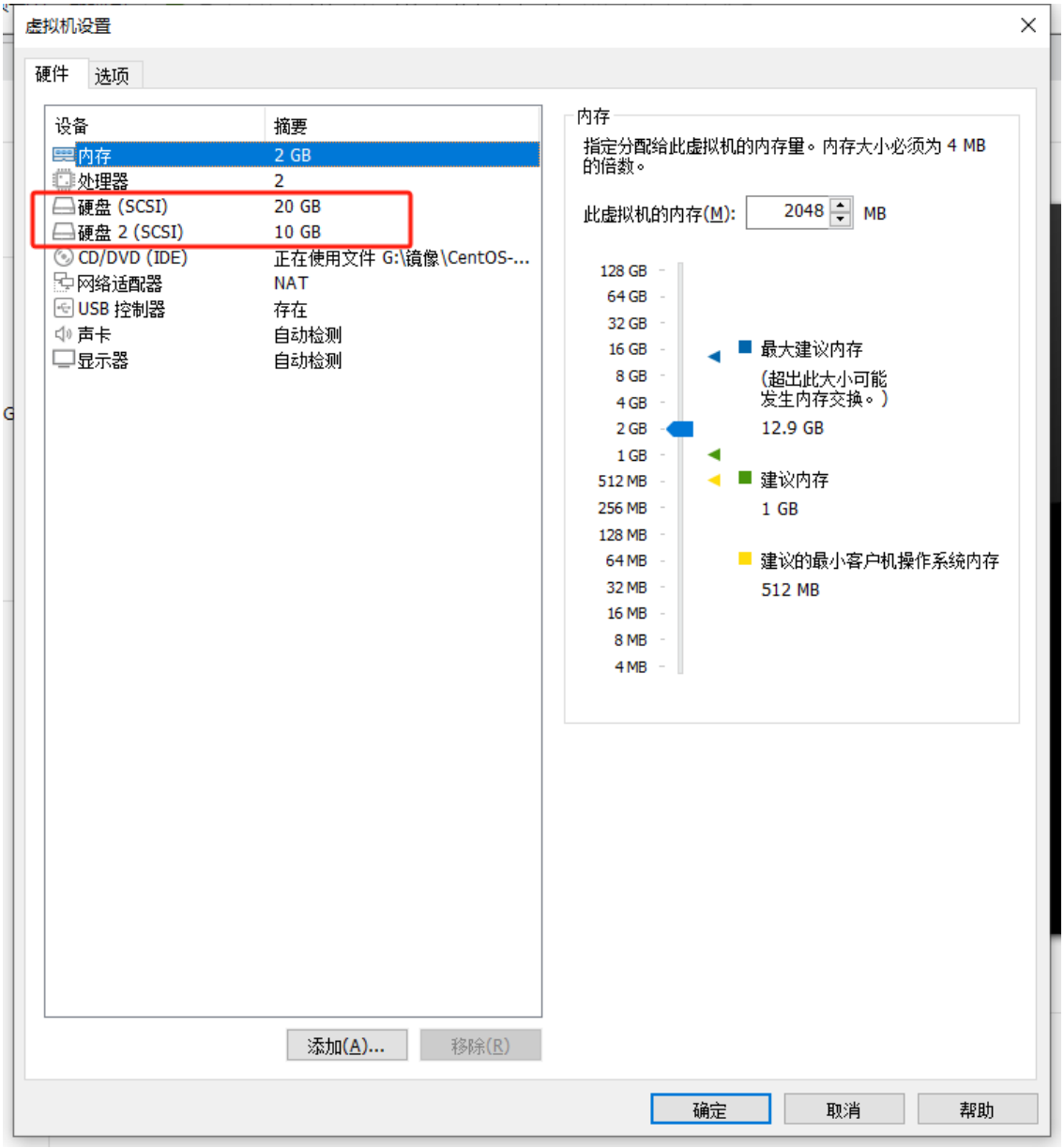
实验环境

- VMware Workstation 虚拟机
- 操作系统：CentOS 8
- 新添加的虚拟磁盘（10GB）

实验步骤

一、在VMware中添加磁盘

1. **关闭虚拟机电源**
2. **编辑虚拟机设置**
    - 点击虚拟机设置 → 添加硬盘 → 默认SCSI类型 → 创建新虚拟磁盘（10GB） → 完成。



3. **启动虚拟机并验证新磁盘**

bash

```
# 查看新磁盘设备（通常为/dev/sdb或/dev/sdc）
[root@localhost ~]# ll /dev/sd*
brw-rw----. 1 root disk 8,  0 Mar 10 19:40 /dev/sda
brw-rw----. 1 root disk 8,  1 Mar 10 19:40 /dev/sda1
brw-rw----. 1 root disk 8,  2 Mar 10 19:40 /dev/sda2
brw-rw----. 1 root disk 8, 16 Mar 10 19:40 /dev/sdb

[root@localhost ~]# fdisk -l
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x618c7ded

Device     Boot   Start      End  Sectors Size Id Type
/dev/sda1  *       2048  2099199  2097152   1G 83 Linux
/dev/sda2       2099200 41943039 39843840  19G 8e Linux LVM
```

```
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/cl-root: 17 GiB, 18249416704 bytes, 35643392 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

二、创建MBR分区

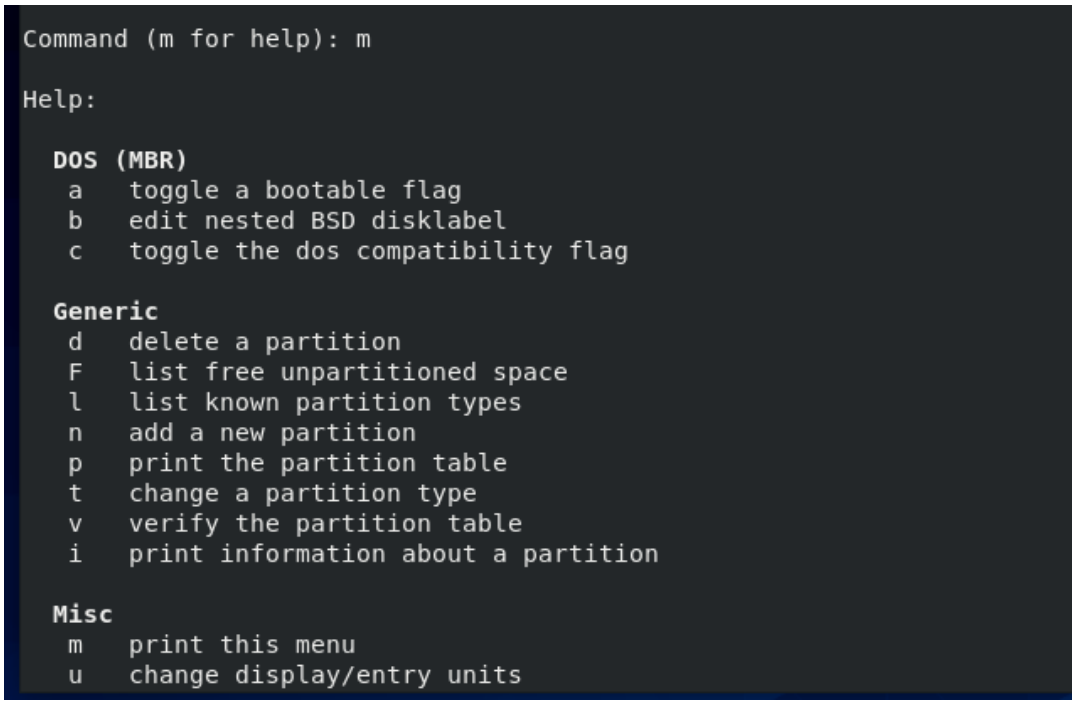
1. **启动分区工具**

   bash

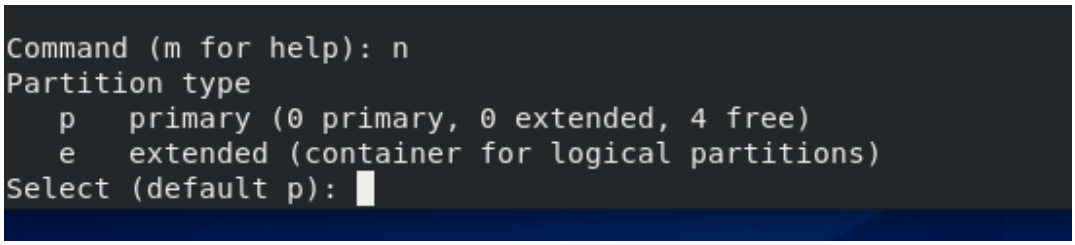
   ```
   fdisk /dev/sdb              #fdisk MBR工具
   ```

2. **创建主分区**

   - **输入** `m` →查看命令帮助

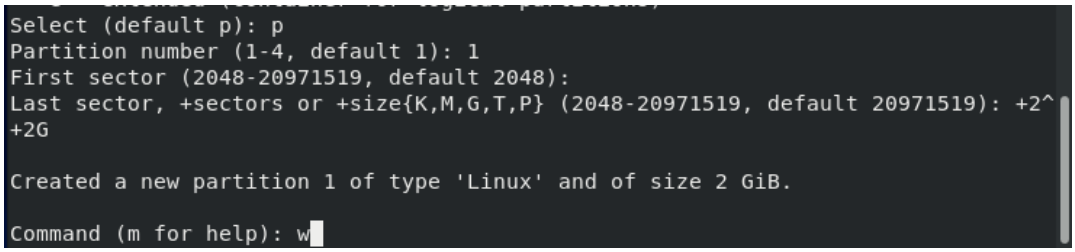
     

   - **输入** `n` → 新建分区

   - **输入** `p` → 选择主分区
   - **输入** `1` → 分区号（1-4）

- **起始扇区按回车**（默认2048）
- **输入** `+2G` → 分配2GB空间
- **输入** `w` → 保存分区表



bash

```
# 分区过程示例
命令(输入 m 获取帮助)：n
Partition type: p (primary)
分区号 (1-4)：1
起始扇区：默认2048
Last 扇区：+2G
命令(输入 m 获取帮助)：w
```

3. **刷新分区表**

   bash

   ```
   [root@localhost ~]# partprobe /dev/sdb
   ```

4. **验证分区**

   bash

   ```
   [root@localhost ~]# fdisk -l /dev/sdb
   Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
   Units: sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disklabel type: dos
   Disk identifier: 0xe7c3a1ef

   Device     Boot Start     End Sectors Size Id Type
   /dev/sdb1        2048 4196351 4194304   2G 83 Linux
   # 输出应包含 /dev/sdb1 分区
   ```

三、创建文件系统（格式化）

bash

```
# 格式化为ext4文件系统
[root@localhost ~]# mkfs.ext4 /dev/sdb1
mke2fs 1.44.6 (5-Mar-2019)
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 56bf0140-1fbe-43c1-b056-d8c45cf4fb56
Superblock backups stored on blocks:
```

```
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# 输出示例：
mke2fs 1.42.9 (28-Dec-2013)
文件系统标签= OS type: Linux ...
```

四、挂载分区

1. **创建挂载点**

   bash

   ```
   mkdir -p /mnt/disk1
   ```

2. **手动挂载**

   bash

   ```
   [root@localhost ~]# mkdir -p /mnt/disk1
   [root@localhost ~]# mount /dev/sdb1 /mnt/disk1
   ```

3. **验证挂载**

   bash

   ```
   [root@localhost ~]# df -hT
   Filesystem          Type      Size  Used Avail Use% Mounted on
   devtmpfs            devtmpfs  886M     0  886M   0% /dev
   tmpfs               tmpfs     904M     0  904M   0% /dev/shm
   tmpfs               tmpfs     904M  9.7M  894M   2% /run
   tmpfs               tmpfs     904M     0  904M   0% /sys/fs/cgroup
   /dev/mapper/cl-root xfs        17G  4.7G   13G  28% /
   /dev/sda1           ext4      976M  143M  767M  16% /boot
   tmpfs               tmpfs     181M  1.2M  180M   1% /run/user/42
   tmpfs               tmpfs     181M  4.6M  177M   3% /run/user/0
   /dev/sr0            iso9660   7.1G  7.1G     0 100% /run/media/root/CentOS-
   8-1-1911-x86_64-dvd
   /dev/sdb1           ext4      2.0G  6.0M  1.8G   1% /mnt/disk1
   [root@localhost ~]#

   # 输出应显示：
   /dev/sdb1 ext4  2.0G  6.0M  1.8G   1% /mnt/disk1
   ```

五、创建扩展分区与逻辑分区

1. **删除原有分区（可选）**

   bash

   ```
   fdisk /dev/sdb
   命令(输入 m 获取帮助)：d → 选择分区号 → w
   ```

2. **创建扩展分区**

   bash

   ```
   fdisk /dev/sdb
   命令：n → e → 默认参数 → w
   ```

3. **创建逻辑分区**

   bash

   ```
   fdisk /dev/sdb
   命令：n → l → 分配空间（如+3G） → w
   ```

4. **格式化并挂载逻辑分区**

   bash

   ```
   mkfs.ext4 /dev/sdb5      # 格式化为ext4
   mkdir /mnt/disk5
   mount /dev/sdb5 /mnt/disk5
   ```

六、永久挂载（重启生效）

1. **编辑 `/etc/fstab` 文件**

   bash

   ```
   vim /etc/fstab
   # 添加以下内容：
   /dev/sdb1  /mnt/disk1  ext4  defaults  0 0
   ```

2. **验证配置**

   bash

   ```
   mount -a  # 自动挂载所有未挂载的分区
   df -h     # 检查是否成功
   ```

七、实验验证

1. **写入测试文件**

   bash

```
echo "Hello, Linux Disk!" > /mnt/disk1/test.txt
cat /mnt/disk1/test.txt
```

2. **重启验证持久性**

   bash

```
reboot
df -hT  # 检查挂载点是否自动加载
```

八、注意事项

1. **分区前备份数据**：误操作可能导致数据丢失。
2. MBR限制：
   - 最多4个主分区，扩展分区只能有1个。
   - 单个分区不超过2TB。
3. **LVM进阶**：继续学习LVM动态调整分区大小。

故障排查

1. **分区未识别**：

   bash

```
partprobe /dev/sdb  # 强制刷新分区表
```

2. **挂载失败**：

   bash

```
dmesg | tail        # 查看内核日志
fsck /dev/sdb1      # 检查文件系统错误
```

将sdb1写满：

```
[root@localhost ~]# dd if=/dev/zero of=/mnt/disk1 bs=1M count=2500
dd: 打开'/mnt/disk1' 失败: 是一个目录
[root@localhost ~]# dd if=/dev/zero of=/mnt/disk1/1.txt bs=1M count=2500
dd: 写入'/mnt/disk1/1.txt' 出错: 设备上没有空间
记录了1900+0 的读入
记录了1899+0 的写出
1991864320 bytes (2.0 GB, 1.9 GiB) copied, 1.91383 s, 1.0 GB/s
[root@localhost ~]# df -hT
文件系统             类型       容量  已用  可用 已用% 挂载点
devtmpfs             devtmpfs   886M     0  886M    0% /dev
tmpfs                tmpfs      904M     0  904M    0% /dev/shm
tmpfs                tmpfs      904M  9.7M  894M    2% /run
tmpfs                tmpfs      904M     0  904M    0% /sys/fs/cgroup
/dev/mapper/cl-root xfs          17G  4.7G   13G   28% /
/dev/sda1            ext4       976M  143M  767M   16% /boot
```

```
tmpfs                tmpfs     181M  1.2M  180M    1% /run/user/42
tmpfs                tmpfs     181M  4.6M  177M    3% /run/user/0
/dev/sr0             iso9660   7.1G  7.1G     0  100% /run/media/root/CentOS-8-1-
1911-x86_64-dvd
/dev/sdb1            ext4      2.0G  1.9G     0  100% /mnt/disk1
```

三、逻辑卷LVM

前言

写满一个磁盘需要几步?

dd if=/dev/zero of=/mnt/disk4/1.txt bs=1M count=2500

```
[root@localhost ~]# dd if=/dev/zero of=/mnt/disk1 bs=1M count=2500
dd: 打开'/mnt/disk1' 失败: 是一个目录
[root@localhost ~]# dd if=/dev/zero of=/mnt/disk1/1.txt bs=1M count=2500
dd: 写入'/mnt/disk1/1.txt' 出错: 设备上没有空间
记录了1900+0 的读入
记录了1899+0 的写出
1991864320 bytes (2.0 GB, 1.9 GiB) copied, 1.91383 s, 1.0 GB/s
[root@localhost ~]# df -hT
文件系统             类型       容量  已用  可用 已用% 挂载点
devtmpfs             devtmpfs  886M     0  886M    0% /dev
tmpfs                tmpfs     904M     0  904M    0% /dev/shm
tmpfs                tmpfs     904M  9.7M  894M    2% /run
tmpfs                tmpfs     904M     0  904M    0% /sys/fs/cgroup
/dev/mapper/cl-root xfs         17G  4.7G   13G   28% /
/dev/sda1            ext4      976M  143M  767M   16% /boot
tmpfs                tmpfs     181M  1.2M  180M    1% /run/user/42
tmpfs                tmpfs     181M  4.6M  177M    3% /run/user/0
/dev/sr0             iso9660   7.1G  7.1G     0  100% /run/media/root/CentOS-8-1-
1911-x86_64-dvd
/dev/sdb1            ext4      2.0G  1.9G     0  100% /mnt/disk1
```

基本磁盘，缺点是无法调整大小！！！

LVM(逻辑卷管理)目的

管理磁盘的一种方式，性质与基本磁盘无异

特点：随意扩张大小

术语

1. 物理卷（PV）→ 砖块

- **比喻**：仓库里的每一块砖头（PV）代表一块物理硬盘或分区。 /dev/sdc3 500G /dev/sdd1 400G /dev/sde 500g
- 特点：
  - 砖块本身不能直接使用，需要组合起来。
- 可以添加更多砖块（硬盘）来扩大容量。

---

2. 卷组（VG）→ 整个仓库 VG 900g

- **比喻**：将许多砖块（PV）堆砌成一个仓库（VG），形成一个统一的存储池。
- 特点：
  - 仓库的大小取决于砖块数量（PV的数量和容量）。
- 可以随时往仓库里添加新砖块（扩展VG）。

---

3. 逻辑卷（LV）→ 仓库里的房间 900g

- **比喻**：从仓库中灵活隔出不同大小的房间（LV），用来存放不同物品（数据）。
- 特点：
  - 房间大小可动态调整（LV扩容/缩容）。
  - 房间可以跨多个砖块（数据可以分布在多个PV上）。

完整比喻场景

1. **建仓库**
   - 你买了几堆砖头（`PV`：`/dev/sdc`、`/dev/sdd`）。
   - 把砖头堆成一个仓库（`VG`：`myvg`）。
2. **隔房间**
   - 在仓库里隔出一个大客厅（`LV`：`mylv1`，500GB）存放家具（系统文件）。
   - 再隔一个小书房（`LV`：`mylv2`，200GB）存放书籍（用户数据）。
3. **动态调整**
   - 如果客厅不够用，可以拆掉一面墙，把仓库的闲置空间（VG剩余容量）合并到客厅。
   - 如果买了新砖头（`PV`：`/dev/sde`），直接添加到仓库（VG扩容），再扩展房间大小。

对比表格

LVM组件	比喻	功能	操作命令
PV	砖块	物理存储单元（硬盘/分区）	`pvcreate`、`pvs`
VG	仓库	整合多个PV的存储池	`vgcreate`、`vgextend`
LV	房间	从VG中划分的逻辑存储空间	`lvcreate`、`lvextend`

实际案例

场景：扩展房间（LV扩容）

1. **初始状态**
   - 仓库（VG）总容量：10块砖头（10TB）。
   - 客厅（LV）当前大小：5TB。
2. **操作步骤**
   - **添加新砖块**：买5块新砖头（`pvcreate /dev/sdf`）。
   - **扩大仓库**：将新砖头加入仓库（`vgextend myvg /dev/sdf`）。

- **扩展客厅**：把仓库的闲置空间（5TB）加到客厅（`lvextend -L +5TB /dev/myvg/mylv1`）。

总结

- **PV是基础**：砖块（硬盘）是仓库的原材料。
- **VG是整合**：仓库的容量和灵活性来自砖块的组合。
- **LV是核心**：房间（逻辑卷）是真正存放数据的地方，可动态调整。

Linux LVM与Swap管理实验手册

实验目标

1. 掌握LVM（逻辑卷管理）的创建与扩展。
2. 学习交换分区（Swap）的配置与管理。
3. 理解动态调整存储空间的原理。

第一部分：LVM逻辑卷管理

一、LVM核心概念

术语	全称	作用	比喻
PV	Physical Volume	物理磁盘或分区	砖块
VG	Volume Group	整合多个PV的存储池	砖块堆成的仓库
LV	Logical Volume	从VG中划分的逻辑分区	从仓库中隔出的房间

二、创建LVM逻辑卷

实验步骤

1. **准备物理磁盘**

   bash

   ```
   # 查看新磁盘（例如/dev/sdc、/dev/sdd）
   ls /dev/sd*
   ```

2. **创建物理卷（PV）**

   bash

   ```
   pvcreate /dev/sdc    #将物理磁盘，转换成物理卷
   # 验证
   pvs          # 查看PV摘要
   pvdisplay    # 查看PV详细信息
   ```

3. **创建卷组（VG）**

bash

```
vgcreate vg1 /dev/sdc      #创建卷组，可以将多个物理卷 创建为一个卷组
# 验证
vgs        # 查看VG摘要
vgdisplay  # 查看VG详细信息
```

4. **创建逻辑卷（LV）**

   bash

   ```
   # 创建10G的LV（命名为mylv）
   lvcreate -L 1G -n lv1 vg1       #逻辑卷的空间基于卷组
   # 验证
   lvs        # 查看LV摘要
   lvdisplay  # 查看LV详细信息
   ```

5. **格式化并挂载LV**

   bash

   ```
   # 格式化为ext4
   mkfs.ext4 /dev/vg1/lv1
   # 创建挂载点并挂载
   mkdir /mnt/mylv
   mount /dev/vg1/lv1 /mnt/lv1
   # 验证
   df -hT /mnt/lv1
   ```

三、扩展卷组（VG）

1. **添加新磁盘（如/dev/sde）并创建PV**

   bash

   ```
   pvcreate /dev/sde
   ```

2. **扩展VG**

   bash

   ```
   vgextend myvg /dev/sde
   # 验证
   vgs
   ```

四、扩展逻辑卷（LV）

1. **扩展LV容量**

   bash

   ```
   # 扩展5G（若VG有空闲空间）
   lvextend -L +5G /dev/myvg/mylv
   ```

2. **调整文件系统大小**

   bash

```
#检测文件系统类型
e2fsck -f /dev/vg1/lv1
# ext4文件系统
resize2fs /dev/myvg/mylv
# xfs文件系统
xfs_growfs /dev/myvg/mylv
#挂载
mount /dev/vg1/lv1 /mnt/lv1
# 验证
df -hT /mnt/mylv
```

第二部分：交换分区（Swap）管理

在Linux系统中，**交换分区（Swap）**是一种用于扩展可用内存的机制，通过将物理内存中不活跃的页面移动到磁盘上的专用空间（交换分区或交换文件），从而优化内存管理并提高系统稳定性

一、Swap的作用

- **内存不足时的后备空间**：防止系统因内存耗尽而崩溃。
- 推荐大小：

内存大小	Swap推荐大小
≤4GB	2×内存
4GB~16GB	4GB
16GB~64GB	8GB
> 64GB	16GB

二、创建Swap分区

实验步骤

1. **创建新分区并标记为Swap**

   bash

```
fdisk /dev/sdf
# 操作：n → p → 1 → 回车 → +2G → t → 82 → w
partprobe /dev/sdf
```

2. **格式化Swap分区**

   bash

```
mkswap /dev/sdf1
```

3. **启用Swap分区**

   bash

   ```
   swapon /dev/sdf1
   # 验证
   free -h
   ```

4. **永久生效（可选）**

   bash

   ```
   echo "/dev/sdf1 swap swap defaults 0 0" >> /etc/fstab
   ```

第三部分：故障排查与命令速查

一、常见问题

1. **挂载失败（wrong fs type）**
   - **原因**：分区未格式化或文件系统损坏。
   - 解决：

     bash

     ```
     mkfs.ext4 /dev/myvg/mylv  # 格式化
     fsck /dev/myvg/mylv       # 修复文件系统
     ```

2. **VG空间不足**
   - **解决**：添加新PV并扩展VG（`vgextend`）。

二、命令速查表

功能	命令
创建PV	`pvcreate /dev/sdX`
创建VG	`vgcreate VG名 PV列表`
创建LV	`lvcreate -L 大小 -n LV名 VG名`
扩展VG	`vgextend VG名 /dev/sdX`
扩展LV	`lvextend -L +大小 /dev/VG名/LV名`
调整文件系统	`resize2fs` 或 `xfs_growfs`
创建Swap	`mkswap /dev/sdX1; swapon /dev/sdX1`

实验总结

- **LVM优势**：动态调整存储空间，避免传统分区限制。
- **Swap配置**：根据内存合理规划，提升系统稳定性。
- 操作要点：
  - 所有操作需谨慎，提前备份数据。
  - 文件系统类型需匹配（如ext4/xfs）。