Linux 文件系统庖丁解牛

相信搞软件开发的同学对文件系统都有一定的了解，即使不是做软件开发工作的同学对文件系统也有感性的认识。其实回忆一下，无论是 Linux 操作系统也好，还是 Windows 或者 Mac 也好，在我们普通用户的视角看到的其实就是一个个文件。比如电影是用视频文件存储，也就是表示某种视频格式的文件；音乐是用音频文件存储的，像 mp3、wave 和 midi 等等格式；图像是用图片文件格式存储的，像 png、jpg 和 bmp 等等。虽然文件中的内容和存储格式不同，但其原理都是一样的，都依赖于文件系统。

我们知道文件系统是位于磁盘之上的，但具体什么原理可能不清楚。也没考虑过为什么不直接使用磁盘。我们先看一下磁盘的结构，磁盘内部如图 1 所示，其内部有若干个盘片，数据存储在磁盘的盘片上。而盘片又划分为磁道和扇区，具体细节本文就不深入分析了。

图 1 磁盘内部结构

如果我们从磁盘上读写数据，从感官上应该是比较复杂的。首先需要知道在哪个盘片上，然后需要知道在盘片的什么位置，然后才能读取或者写入数据。实际上不用那么复杂，磁盘的控制器已经替我们做了很多事情，它对这些内部的结构进行了统一管理，呈现给我们的只是一个线性的地址。比如一个 1T 的硬盘，其呈现给我们的就是从 0 字节开始，以 1 字节递增，直到 1TB 的地址空间。当然，机械磁盘可进行读写的最小粒度为 1 个扇区（512 字节）

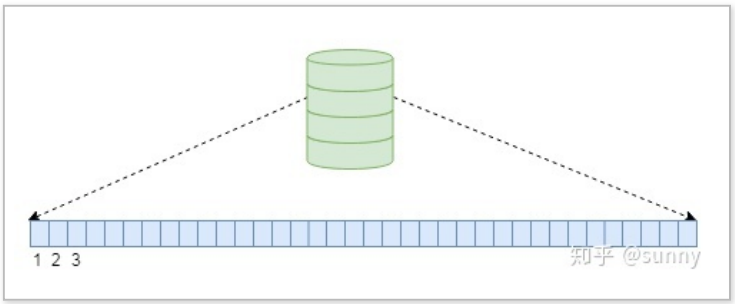

图 2 磁盘存储空间抽象

既然可以这么方便的访问磁盘空间了，那为什么还要文件系统呢?

其实最主要的原因有 3 方面，也即:

- 便于磁盘空间的管理
- 方便数据的组织和查找
- 提高磁盘空间的使用率

便于磁盘空间管理

我们不考虑存储操作系统的磁盘，即使是存放普通数据（例如放电影视频文件）的磁盘，如果没有文件系统会是什么样子。

比如我们把《空中客车》放到 0 到 1GB 的空间，《蝙蝠侠》放到 1G 到 3G 的空间，《蜘蛛侠》放到 3G 到 9G 的空间等等。然后呢? 我们还的找个地方记住这些电影的名称和电影存储起始位置和长度这些信息，否则我们就找不到我们想要的电影了。

再比如我们不想要《空中客车》这部电影了，那么这部分空间就可以存储其它电影了。比如我们有一部《异形》，大小是 1.5G，这时显然没法放到《空中客车》原来的位置，因为这个空间只有 1G，因此之后放到《蜘蛛侠》后面。好嘛，简直难以想象，经过几百次添加删除后磁盘会变成什么样子。而且

我们还得用个本子也好，或者什么也好记录每个电影的名称、位置还有磁盘的可用空间。

而如果有了文件系统之后（格式化后）呢？我们只需要建立文件夹（当然也可以不创建），让文件拷贝到里面就行了。我们根本不用考虑磁盘上的数据是怎么管理的。

方便数据的组织和查找

先感受一下在 Linux 操作系统下文件的组织形式，通常是一个树状的结构。也就是磁盘被格式化后通常用户会创建若干个文件夹，然后在文件夹中再创建文件夹或者存储文件。

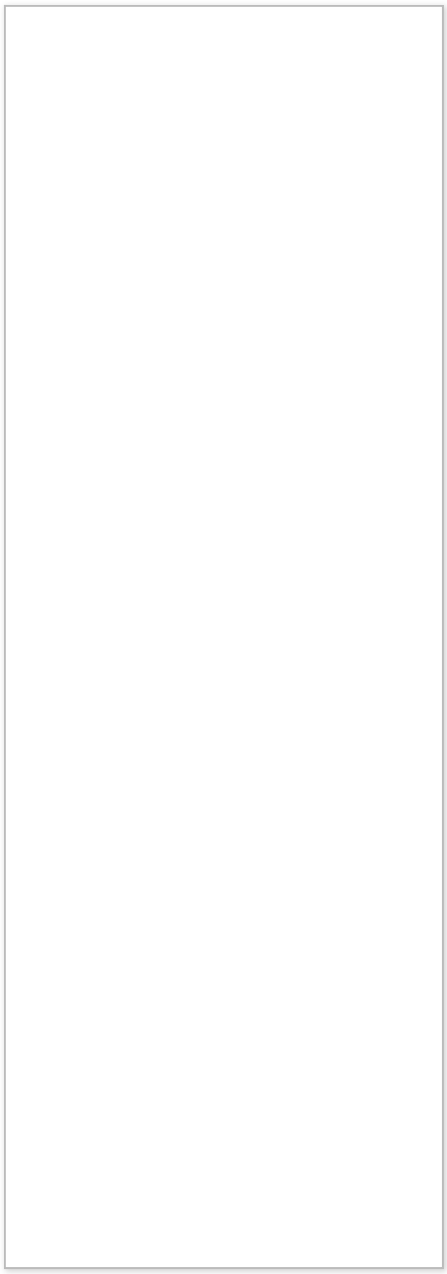

图 3 文件目录树

比如我们用一个磁盘来专门存储数据，格式化之后我们创建若干个文件夹，分别是“电影”、“音乐”、“照片”和“电子书”等。然后在电子书里面有分别创建“Linux”、“编程语言”、“历史”和“小说”等等。这样我们将所有数据组织成非常有条理的树形结构。为了形象，我们画成如图 4 的样子，可以看出通过文件系统使我们对数据有了很清晰的规划，也很方便后续查找我们想要的数据。

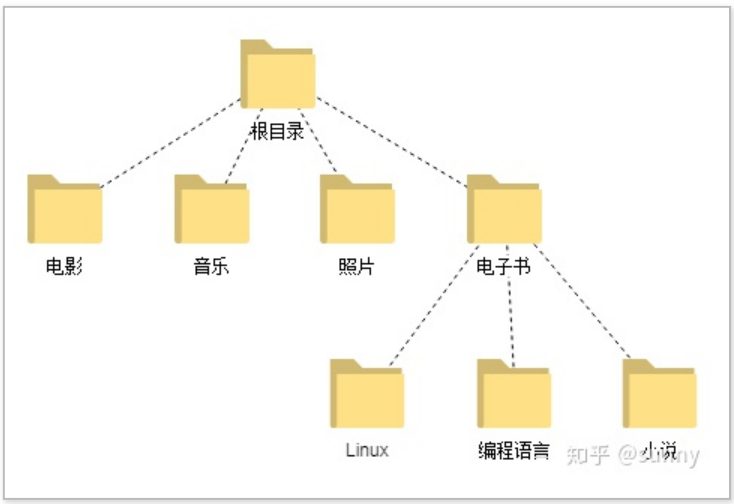

图 4 目录树示意图

提高磁盘空间的使用率

如前文所说，如果没有文件系统，不需要的文件的空间再利用就会非常麻烦。有可能这块空间的大小是 1G 或者 1M，而新数据的大小是 2G。那这个空间就无法使用。频繁的释放和使用空间之后，可能会留下很多小空间（空洞），而无法被使用，这样就造成磁盘空间的极大浪费。

使用文件系统之后，文件系统会将磁盘空间切割为比较小的存储单元（例如 4K 或者 8K 等）进行管理。如果出现释放空间产生空洞的情况，文件系统内部会进行空洞和数据的交换，从而生成比较大块的可用磁盘空间。这样从整体来说就极大地提升了磁盘的整理使用率。

Linux 的文件系统

目前在 Linux 操作系统中支持很多种文件系统，包括 Ext2、Ext4、Btrfs 和 XFS 等，多达几十个文件系统。虽然支持的文件系统种类很多，但从用户层面使用方式无任何差别，用户并不感知其中的差异。对于普通用来说，数据都是组织成上文所述的树状结构。那么这种方式是如何实现的呢?

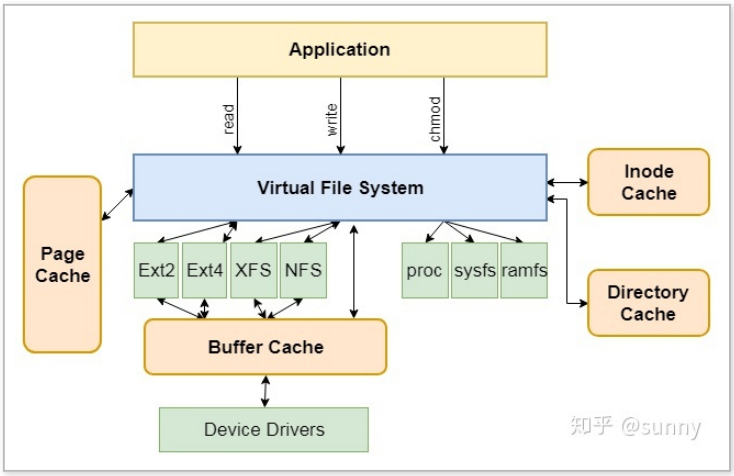

图 5 Linux 的虚拟文件系统

Linux 操作系统对各种文件系统的支持是通过名为 VFS 的组件实现的，也就是虚拟文件系统（Virtual File System）。如图 5 所示，VFS 作为一个抽象层，为用户提供统一的接口，屏蔽了其它具体文件系统（例如 Ext4 和 Btrfs 等）的实现。VFS 为用户提供了 open、close、read 和 write 等接口。

说了半天，那么文件系统到底是怎么管理磁盘，将磁盘空间转换为我们看到的文件夹和文件的呢？其具体方法就是把磁盘划分为一个个的小块，就像切豆腐一样。然后把磁盘划分为不同的功能区，比如元数据区和数据区。而元数据区其实实现对磁盘空间的管理，就好像前文说的账本，里面记录着哪些磁盘空间被使用，哪些磁盘空间已经被占用。

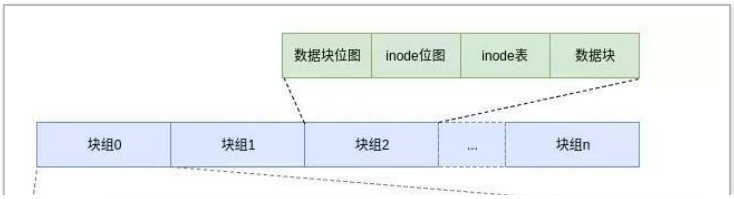

图 6 Ext4 磁盘布局图

经过文件系统的管理之后，文件内的数据就被映射到磁盘上的一块块的空间。而文件和磁盘空间的关系由文件系统管理，不需要用户操心。如图 7 所示，某个文件被映射到磁盘中的 3 个不同的空间。

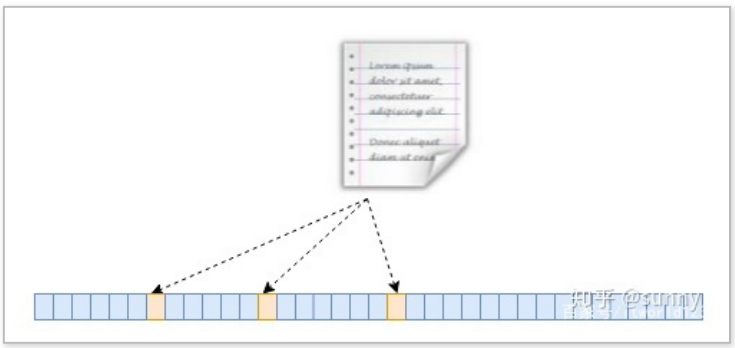

图 7 文件映射与磁盘空间映射

当然，实际的映射关系比可能比上图要复杂得多，但基本原理是这样。用户关心的只是文件名称和路径，而其存储的数据则有文件系统管理。当然，每个文件系统对数据的组织形式是不同的，以 Ext2 文件系统为例，其形式图 8 所示，其通过一些磁盘指针的方式记录了文件数据的存放位置，这样当用户读取数据时，文件系统根据数据的偏移地址和其记录的对应关系就可以找到数据具体存储在磁盘的什么位置，并进行读取。

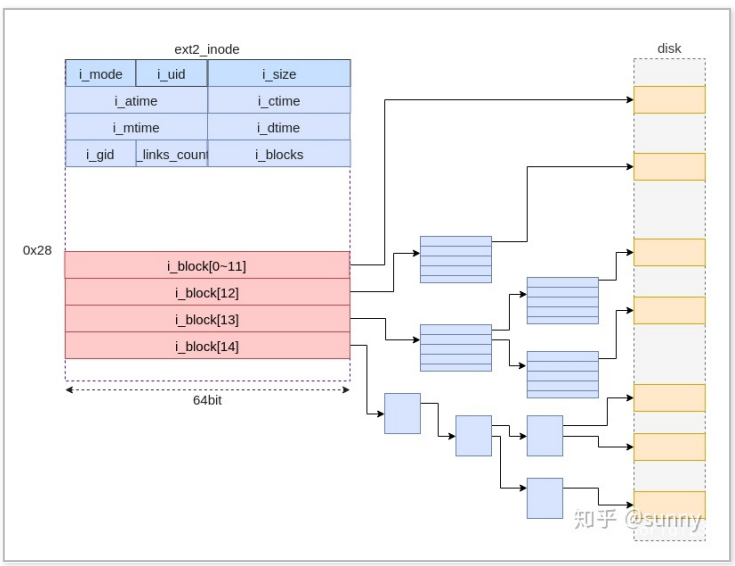

图 8 Ext2 间接块方式

好了，本文就介绍到这，具体文件系统的实现细节请参考号的历史文章。有任何问题或者不同意见，请在下面留言。

写下你的评论...

很想问下，这种示意图是用什么软件画出来的？很好看啊

drawio，你自己上网搜一下

别问了，这不是他写的

可以讲讲 mount 时发生的事情，结合具体命令更清晰

有时间写一下

这张虚拟文件系统的图是多少年前的啊 [捂脸]

文章写的不错，我更关心那几个 cache，他们是如何有效工作的，文件系统的效率，全靠它们，还有突然断电，是如何保护系统的，关键的关键。

谢谢，后面写一下

建议看鸟哥的书，很详细

全文完