Linux磁盘和文件系统简介

文件系统：存储设备上存储数据的方式方法
磁盘主要由盘片、机械手臂、磁头和主轴马达组成，而数据的写入实际是写在盘片上，磁盘的最小存储单位为扇区，每个扇区为512字节，扇区组成一个圆就叫做磁道，而多个盘片的硬盘同一位置的磁道组成柱面

存储容量 = 磁头数 × 磁道(柱面)数 × 每道扇区数 × 每扇区字节（512字节）

Linux当中，硬盘是以挂载形式存在的，挂载就是利用一个目录当成进入点，将磁盘分区的数据放在该目录下，进入该目录即进入该分区的意思。

文件系统有：NTFS（New Technology File System）新技术文件系统、Windows专用；FAT、ext系列、xfs等
RHEL5当中默认文件系统为ext3、RHEL6当中默认文件系统为ext4、RHEL7当中默认文件系统为xfs。

我们在前面的课程中讲到过，每一个磁盘都有MBR，位于0,0,1的位置，存放了引导微代码以及磁盘分区表和校验值，所以我们来看流程，当我们安装一块硬盘的时候，首先先应该要做的就是分区，分区表大小只有64字节，所以主分区+扩展分区最多能创建4个，而且扩展分区能且只能创建一个，我们学习了硬盘的物理结构，应该对硬盘有一个认识了。
而分区其实就是告诉操作系统在此硬盘中可以访问的区域是由A柱面到B柱面之间的块，也就是说，磁盘分区就是指定分区的起始与结束柱面。
磁盘分区的目的：方便管理磁盘文件

磁盘在分完区后需要进行格式化，格式化就是为了遵循该系统的存储数据的方式，按照该方式进行写入数据，其实格式化就是为了将文件系统写入到磁盘当中，让磁盘变成能被操作系统所识别使用的系统格式

文件系统的运行是与操作系统的文件数据有直接关系的，比如RHEL6.5所用的文件系统为ext4，该文件系统中的文件数据除了文件实际内容外还包含了很多属性，例如Linux操作系统的文件权限rwx和文件属性属主属组时间参数等，文件系统通常将这两部分数据分别存放在不同的块，权限和属性放置到**inode**中，至于实际数据则放到**data block**块中，而且还有一个超级块（**superblock**）会记录整个文件系统的整体信息，包括inode与block的总量、使用量、剩余量以及文件系统的格式等很多参数信息。

inode：记录文件的属性，一个文件占用一个inode，同时记录此文件的数据所在block号
block：实际记录文件的内容，如果文件太大，会占用多个block
superblock：记录此文件系统的整体信息
每个inode和block都有唯一编号，而且每个文件都会占用一个inode，inode中有文件真实数据所在的block号，所以说，当我们找到一个文件的inode时，同时也就可以通过inode查找到该文件真实数据所在的block位置，即能够读取到该文件的真实数据，这样一来我们硬盘上的数据在短时间内就都能够被读取出来，读取的性能就比较好

包括磁盘碎片的整理，都是通过将数据真实存放在的block进行整理

当我们的硬盘非常大，到达TB级别的时候，分区后inode和block的数量是非常巨大的，而且除非在修改文件系统的情况下，inode和block是不会再发生变动的，所以庞大的数据量会导致很难管理，所以其实在我们进行格式化的时候，都是分为多个块组的，每一个块组都有独立的inode、block和superblock，而且在每一个分区当中，我们在将开机启动流程的时候有提到过我们每一个系统需要指定的文件系统且在不同分区，而每一个分区的文件系统中最前面有一个启动分区，就是我们说过的boot loader，这个boot loader可以安装我们的操作系统引导装载程序，所以才有了我们的多引导环境。

那么我们来详细讲解一下上面的各个参数

block是存放数据的地方，在我们进行格式化的时候，会设定一个“分配单元大小”，这个值决定了我们block的大小，而这个值也就决定了我们能够存放的单一文件大小以及我们最大文件系统总的容量，”分配单元大小“越小越节省空间，“分配单元大小”越大越浪费空间。也就是说block的大小和数量在格式化之后就不能够再改变了（除非重新格式化）；每一个block当中只能存放一个文件的数据；如果文件大小大于block的大小，则一个文件占用多个block数量；如果文件大小小于block的大小，则该block的剩余空间就不能再使用了

inode是存放文件属性及block块位置等信息的，所以肯定是以表的形式存在的，在inode当中至少记录了：文件的权限rwx；文件的属主属组；文件的大小；文件的atime、ctime、mtime；文件特性标志，如Set位等；文件实际数据的位置等信息。

inode的大小固定为128字节；每一个文件只会占用一个inode；所以，我们能创建的文件数量与inode的数量有关；系统读取文件的时候先找inode，并查询inode中记录的权限与用户是否符合，符合才继续读取文件的真实内容。

我们来看inode和block的关系，inode当中记录了非常多的信息，而inode固定大小为128字节，单记录一个block位置需要占用4字节，当inode记录的文件A大小为1GB时且block块大小为4k时，A文件就会占用十多万个block，而inode当中也需要记录十多万的block位置，就会出现冲突，为了解决这一冲突，系统将inode记录block号码的区域定义为12个直接、1个间接、一个双间接、和一个三间接记录。

这种情况下，我们通过inode能指定的block的数量（block大小以1k为例）如下：

12个直接指向：12*1K=12K \\由于是直接指向，所以可以有12条记录

间接：256*1K=256K \\每个block记录占4字节，因此1K能指定256个记录

双间接：256*256*1K=256^2K \\第一层的指向有256个记录指向二层，二层每个记录能指定256个

三间接：256*256*256*1K=256^3K \\同上

总额为 12+256+256*256+256*256*256（K）=16GB

所以当文件系统将block大小定义为1k时，单个文件大小最大为16G

这个计算方法不能用在2k和4k等更大上，大于2k的block会受到文件系统自身的限制，所以计算结果会有偏差。

superblock是记录整个文件系统相关信息的地方，没有超级块，就没有文件系统了，它主要记录了：block与inode的总量；未使用和已使用了的block和inode数量；block和inode的大小；文件系统挂载的时间、最近一次写入数据的时间等相关信息；以及一个文件系统是否被挂载的数值validbit，已挂载为0，未挂载为1。

当我将一个分区进行格式化之后，文件系统为ext4，（200G），假设分成了10个块组block group，即有10个superblock，但是ext机制让系统在读取该分区的时候首先读取第一个块组当中的超级块（后面的块组没有超级块），来确定该分区的文件系统