一、inode是什么？

理解inode，要从文件储存说起。

文件储存在硬盘上，硬盘的最小存储单位叫做"扇区"（Sector）。每个扇区储存512字节（相当于0.5KB）。

操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。**这种由多个扇区组成的"块"，是文件存取的最小单位**。"块"的大小，最常见的是4KB，即连续八个 sector组成一个 block。

文件数据都储存在"块"中，那么很显然，我们还必须找到一个地方储存文件的元信息，比如文件的创建者、文件的创建日期、文件的大小等等。这种储存文件元信息的区域就叫做inode，中文译名为"索引节点"。

二、inode的内容

inode包含文件的元信息，具体来说有以下内容：

　　\* 文件的字节数

　　\* 文件拥有者的User ID

　　\* 文件的Group ID

　　\* 文件的读、写、执行权限

　　\* 文件的时间戳，共有三个：ctime指inode上一次变动的时间，mtime指文件内容上一次变动的时间，atime指文件上一次打开的时间。

　　\* 链接数，即有多少文件名指向这个inode

　　\* 文件数据block的位置

**可以用stat命令，查看某个文件的inode信息：**

**stat example.txt**

**stat -f .**

**stat -f .**查看文件系统的状态，比如Block size等； stat filename命令输出的blocks单位通常为512bytes，也就是一个扇区大小，而系统文件存取为一个Block size大小，例子：

[root@localhost home]# stat mem.cc

File: ‘mem.cc’

Size: 190 Blocks: 8 IO Block: 4096 regular file

Device: fd00h/64768d Inode: 134296197 Links: 1

Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)

Context: unconfined\_u:object\_r:home\_root\_t:s0

Access: 2017-02-26 07:22:45.693165538 -0500

Modify: 2017-02-20 18:53:38.514058720 -0500

Change: 2017-02-26 07:23:33.104500015 -0500

Birth: -

mem.cc大小为190字节，这里Blocks为扇区，512bytes，但是文件会占用一个Block大小，4096bytes，故Blocks为8。

小结：

1. stat命令输出的Blocks单位通常是512bytes，也就是一个扇区。

2. 一个文件假设只有几个字节，其实也会占用一个文件块(Block size)大小，通常是4096bytes。

3. 系统通常一次会读取一个Block size大小，而不是一个扇区大小。

总之，除了文件名以外的所有文件信息，都存在inode之中。至于为什么没有文件名，下文会有详细解释。

三、inode的大小

inode也会消耗硬盘空间，所以硬盘格式化的时候，操作系统自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是**inode区（inode table）**，存放inode所包含的信息。

每个inode节点的大小，一般是128字节或256字节。inode节点的总数，在格式化时就给定，一般是每1KB或每2KB就设置一个inode。假定在一块1GB的硬盘中，每个inode节点的大小为128字节，每1KB就设置一个inode，那么inode table的大小就会达到128MB，占整块硬盘的12.8%。

查看每个硬盘分区的inode总数和已经使用的数量，可以使用df命令。

df -i

查看每个inode节点的大小，可以用如下命令：

sudo dumpe2fs -h /dev/hda | grep "Inode size"

dumpe2fs - dump ext2/ext3/ext4 filesystem information，只能查看ext的文件系统，可以用blkid命令确认一下分区的文件系统类型，命令如下：

blkid /dev/sda1

/dev/sda1: UUID="5876faf5-a7dc-49f5-8182-77ee36a94f50" TYPE="xfs"

可以看到文件系统为xfs。

或者用mount -l命令。

由于每个文件都必须有一个inode，因此有可能发生inode已经用完，但是硬盘还未存满的情况。这时就无法在硬盘上创建新文件。

四、inode号码

每个inode都有一个号码，操作系统用inode号码来识别不同的文件。

这里值得重复一遍，Unix/Linux系统内部不使用文件名，而使用inode号码来识别文件。对于系统来说，文件名只是inode号码便于识别的别称或者绰号。表面上，用户通过文件名，打开文件。实际上，系统内部这个过程分成三步：首先，系统找到这个文件名对应的inode号码；其次，通过inode号码，获取inode信息；最后根据inode信息，找到文件数据所在的block，读出数据。

使用ls -i命令，可以看到文件名对应的inode号码：

ls -i example.txt

五、目录文件

Unix/Linux系统中，目录（directory）也是一种文件。打开目录，实际上就是打开目录文件。

目录文件的结构非常简单，就是一系列目录项（dirent）的列表。每个目录项，由两部分组成：**所包含文件的文件名，以及该文件名对应的inode号码**。

ls命令只列出目录文件中的所有文件名：

ls /etc

ls -i命令列出整个目录文件，即文件名和inode号码：

ls -i /etc

如果要查看文件的详细信息，就必须根据inode号码，访问inode节点，读取信息。ls -l命令列出文件的详细信息。

ls -l /etc

**文件权限后面就是链接数。**

六、硬链接

一般情况下，文件名和inode号码是"一一对应"关系，每个inode号码对应一个文件名。但是，**Unix/Linux系统允许多个文件名指向同一个inode号码**。这意味着，可以用不同的文件名访问同样的内容；对文件内容进行修改，会影响到所有文件名；但是，删除一个文件名，不影响另一个文件名的访问。这种情况就被称为"硬链接"（hard link）。

ln命令可以创建硬链接：

ln 源文件 目标文件

运行上面这条命令以后，源文件与目标文件的inode号码相同，都指向同一个inode。inode信息中有一项叫做"链接数"，记录指向该inode的文件名总数，这时就会增加1。反过来，删除一个文件名，就会使得inode节点中的"链接数"减1。当这个值减到0，表明没有文件名指向这个inode，系统就会回收这个inode号码，以及其所对应block区域。

**“硬链接”本质上是“目录项”的同义词**。当一个目标第一次被创建，就会为它创建一个目录项，这其实就是硬链接，而大多数人常常把“硬链接”联想成“为一个已有的对象创建一个额外的目录项”。但是原来的目录项其实没有任何特殊，所有的链接都是平等的，所以一定意义上来说没有方法能识别出哪个是原来的。

目录也可以包含目录，当然这是通过硬链接完成的。当一个子目录被创建时，在其父目录中也创建了一个目录项，这个目录项用于将子目录的名称与新创建inode关联起来。此外，新的目录文件中也自动创建了两个目录项，这两个目录项分别将"."，".." 和当前目录及其父目录关联起来。所以，创建一个子目录会创建一个新的硬链接到其父目录，以及两个对新创建对象（子目录）的硬链接：一个来自其父目录，另一个来自他自己（"."），也就是说一个目录项的硬链接数最少是2。

**目录硬链接比较特殊。首先，创建它们惟一的方法是创建目录**；操作系统硬链接函数不会允许一个硬链接的操作目标是一个目录inode。其中的原因是可能会在文件系统目录结构中产生循环。这也得根据内核是否允许目录硬链接也需要遵从文件系统模块本身。

这里顺便说一下目录文件的"链接数"。创建目录时，默认会生成两个目录项："."和".."。前者的inode号码就是当前目录的inode号码，等同于当前目录的"硬链接"；后者的inode号码就是当前目录的父目录的inode号码，等同于父目录的"硬链接"。所以，任何一个目录的"硬链接"总数，总是等于2加上它的子目录总数（含隐藏目录，这里的2是父目录对其的“硬链接”和当前目录下的".硬链接“）。

七、软链接

除了硬链接以外，还有一种特殊情况。文件A和文件B的inode号码虽然不一样，但是文件A的内容是文件B的路径。读取文件A时，系统会自动将访问者导向文件B。因此，无论打开哪一个文件，最终读取的都是文件B。这时，文件A就称为文件B的"软链接"（soft link）或者"符号链接（symbolic link）。

这意味着，文件A依赖于文件B而存在，如果删除了文件B，打开文件A就会报错："No such file or directory"。这是软链接与硬链接最大的不同：文件A指向文件B的文件名，而不是文件B的inode号码，文件B的inode"链接数"不会因此发生变化。

ln -s命令可以创建软链接。

ln -s 源文文件或目录 目标文件或目录

八、inode的特殊作用

由于inode号码与文件名分离，这种机制导致了一些Unix/Linux系统特有的现象。

1. 有时，文件名包含特殊字符，无法正常删除。这时，直接删除inode节点，就能起到删除文件的作用。

根据inode删除文件：

**find -inum [inode数字] -exec rm -i {} \;**

　　2. 移动文件或重命名文件，只是改变文件名，不影响inode号码。

　　3. 打开一个文件以后，系统就以inode号码来识别这个文件，不再考虑文件名。因此，通常来说，系统无法从inode号码得知文件名。

第3点使得软件更新变得简单，可以在不关闭软件的情况下进行更新，不需要重启。因为系统通过inode号码，识别运行中的文件，不通过文件名。更新的时候，新版文件以同样的文件名，生成一个新的inode，不会影响到运行中的文件。等到下一次运行这个软件的时候，文件名就自动指向新版文件，旧版文件的inode则被回收。

九 实际问题

在一台配置较低的Linux服务器（内存、硬盘比较小）的/data分区内创建文件时，系统提示磁盘空间不足，用df -h命令查看了一下磁盘使用情况，发现/data分区只使用了66%，还有12G的剩余空间，按理说不会出现这种问题。 后来用df -i查看了一下/data分区的索引节点(inode)，发现已经用满(IUsed=100%)，导致系统无法创建新目录和文件。

查找原因：

　　/data/cache目录中存在数量非常多的小字节缓存文件，占用的Block不多，但是占用了大量的inode。

解决方案：

　　1、删除/data/cache目录中的部分文件，释放出/data分区的一部分inode。

　　2、用软连接将空闲分区/opt中的newcache目录连接到/data/cache，使用/opt分区的inode来缓解/data分区inode不足的问题：

　　ln -s /opt/newcache /data/cache