元数据metadata：任何文件系统中的数据分为数据和元数据。数据是指普通文件中的实际数据，而元数据指用来描述一个文件的特征的系统数据，诸如访问权限、文件拥有者以及文件数据块的分布信息等等。（这个元数据的定义还是不够详细，另外，元数据到底怎么存储和读取的，在linux系统下，元数据怎么查看）

Inode索引节点：inode包含文件的元信息（元数据与inode的区别），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

总之，除了文件名以外的所有文件信息，都存在inode之中。

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。（文件名如何和inode节点对应，文件名又是怎么存储的？）

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。（目录文件应该也对应相关的inode，那么这个inode又是怎么处理的）

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）。

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“。（感觉理解起来有点问题，所有子目录都有一个..是对当前目录的硬链接，然后该目录下的.又是一个当前目录的硬链接，还差一个？？？应该是绝对路径本身也是一个硬链接。如果在A地建立了一个文件夹，在B地建立一个硬链接，那么删除A，链接B的绝对地址还有意义吗？）

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）。（那么自己手动建立一个文件，里面的内容是其他文件的路径，是不是就表明这是一个软链接？还有如何查看软链接文件的内容？）

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。

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。

　　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。（也就是说同样的文件名可以对应不同的inode节点，那么文件名和inode节点之间的对应关系到底是怎么样的？换句话说，如何从文件名找到inode节点，以及如何从inode节点找到文件名？或者，是否需要去寻找这个文件名？）

硬链接是存在同一个文件系统中，而软链接却可以跨越不同的文件系统。

**软链接：**

1.软链接，以路径的形式存在。类似于Windows操作系统中的快捷方式

2.软链接可以 跨文件系统 ，硬链接不可以

3.软链接可以对一个不存在的文件名进行链接

4.软链接可以对目录进行链接

**硬链接:**

1.硬链接，以文件副本的形式存在。但不占用实际空间。

2.不允许给目录创建硬链接（硬链为什么不能链接到目录）

3.硬链接只有在同一个文件系统中才能创建（会在你选定的位置上生成一个和源文件大小相同的文件（对吗））

软链接产生了一个文件，文件中是链接对象的地址，那么硬链接产生了什么？

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（一直没有理解，目前暂时能够想到的一个合理的解释是把源文件链接到目标文件后，目标文件新产生的数据实际存储在源文件中）

那么inode节点和文件到底是怎么存储的，和什么相关？inode节点的存储和文件目录的结构有什么关系？

软链接有着自己的 inode 号以及用户数据块。因此软链接的创建与使用没有类似硬链接的诸多限制：