## CSE AlphaFileSystem 思路

1. **关于Id**

系统中出现了两种类型的ID，一种是字符串，一种是整形。大部分也许会这么实现，写两个类（或接口），实现（或继承）这个接口，如StringId和LongId。这是对的！内部的实现大家也许会这么写，StringId有一个私有属性（类型为String），LongId有一个私有属性（类型为long）。那写什么方法来获得这些私有属性呢？**建议toString方法**，这个方法继承自Object，好处是不需要在Id中添加其他方法来实现多态。另外，我们的LongId应该会提供一个方法，返回一个新的未分配的Id。可能大家会选择静态属性，自增，对的！我建议不要从0开始自增（每次程序起来之后都会从0开始自增，有可能会冲突），**建议从当前时间开始自增**，这样就相当于实现了uuid（并行的时候仍有风险）。

1. **关于File**

系统要实现File接口。首先需要明白它指的是什么？对比java的类，可以发现我们要实现的**File等效于java的File加上FileInputStream和FileOutputStream**！！！（有意思吧）



**这里的File只是一个句柄，创建一个对象不代表创建了一个新的文件，只是创建了一个新句柄!!!通过这个句柄可以访问和修改它所对应的内容。**

这是我通过调用AplhaFileSystem接口实现的copy（上图）。**File接口建议增加方法：exists()（File当一个句柄看待）**。当文件存在时，我们对文件进行读写，这个方法的实现只需要检查相应的meta文件是否存在就行了。**File的read**: 首先根据游标和文件长度确定实际读取的字节数量，再通过游标和实际数量找到影响的逻辑块编号，通过逻辑块找到物理块句柄（也就是实际block文件所对应的句柄，即Block对象），再通过这个句柄的相应方法读出整个块，再从整个块中把需要的部分数据提取出来。**File的write**：首先是根据游标和写入的字节长度找到影响的逻辑块编号，产生对应数量的新的逻辑块（考虑到副本，对应的物理块会更多），**写的时候记得要拷贝出部分数据到新的物理块中**，再用新的逻辑块替换掉旧的逻辑块（这个地方，逻辑块实际上只是FileMeta里面的一行内容而已！！！）**File的move:** 移动文件的游标，不详述，可以看lseek。**注意上面的文件读写都要考虑游标并且移动游标！！！**

1. **关于block**

这里的block指的是物理块（.meta和.data文件）的句柄，**它只是一个句柄，创建一个对象不代表创建了一个新的物理块，只是创建了一个新句柄。**

Block的方法实现起来都容易，无非是读文件，实例化某些变量，但是我不建议创建对象的时候就去就去读文件，去初始化这些内容，而是在访问到相关属性的时候去读取（由于物理块是不变的，实际中可以通过属性缓存部分数据）。

1. **关于manager**

这个地方将两个manager合在一起讲是因为我感觉他们是差不多的，都是**用来创建句柄的。**先看FileManger，可能会这么是实现，FileManger一实例化，就创建一个list，把所有的file对象创建好，放到这个列表里面。GetFile时就在里面取，newFile时就创建file放进去，再返回。这样实现挺好的，但是比较浪费内存，很多file可能不会用到。**我建议大家可以这样实现：getFile直接返回新的一个File对象，newFile创建一个新的File对象，初始化为空文件，返回。**可能会诧异！！！getFile为什么返回了一个新的file？！！！**有两点：一，File只是文件句柄，创建File对象不是拷贝了一次文件（这个和java的File类似）；二，程序的扩展性提高，那么就容易实现判断一个文件是否存在。这样不浪费内存吗，用一次new一次？答：对象会很快被释放掉。**那么有人可能会关注File的构造函数做了什么？答：确定了自己的名字，其他什么都没有做。于是为了唯一确定这个文件，还需要确定它的blockManger。

BlockManger的相应方法和FileManger的相应方法类似，原因也类似。稍显不同的一点是，BlockManger的newBlock会传入byte数组，而FileManger的newFile方法只需要传入一个Id（本质上是文件的名字而已）。这个不同导致的是，newBlock时在文件（.meta和.data）中存入了一些数据，而newFile时在文件（.meta）存入的时一些默认的“空数据”。

拙见而已，见笑私拆。闲暇之余瞥见，若有所获，吾当欣慰。