1. Linux中包括两种链接：硬链接(Hard Link)和软链接(Soft Link),软链接又称为符号链接（Symbolic link）。

* **硬链接**：硬链接是文件的别名。从技术上讲,他们公用一个inode(inode中包含了一个文件的所有必要的信息，说inode就是文件也是没有问题的)。 由于linux下的文件是通过索引节点（Inode）来识别文件，硬链接也可以认为是一个指向文件索引节点的指针，系统并不为它重新分配inode,每添加一个硬链接，文件的链接数就加1, 删除一个则链接数减1。

大多数系统不允许创建一个目录的硬链接。在允许创建目录硬链接的系统上，只有超级用户才能这样做。

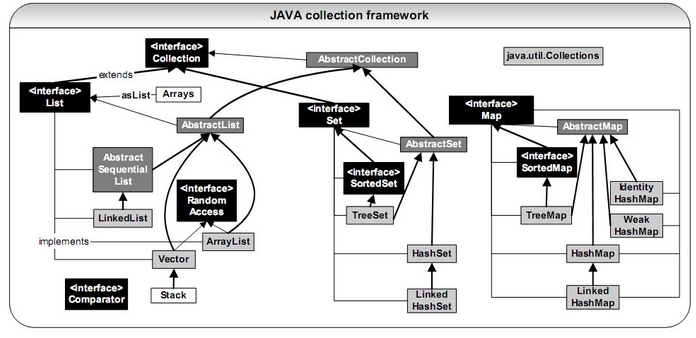
不能创建跨文件系统(分区)的硬链接（这些限制在POSIX中都不是强制性的）。

* **软链接**：软链接是一种特殊的文件类型,其中包含对另一个 文件/目录 以 绝对/相对 路径形式的引用.软链接可以看做是对一个文件的间接指针,相当于windows下的快捷方式。软链接没有任何文件系统的限制，任何用户可以创建指向 文件/目录 的符号链接。甚至可以跨越不同机器、不同网络对文件进行链接。创建文件的软链接时，软链接会使用一个新的inode，所以软链接的inode号和文件的inode号不同(表明他们是两个不同的文件)，软链接的inode里存放着指向文件的路径，删除源文件，软链接也无法使用了，因为文件的路径不存在了；删除软链接对原文件没有任何影响。当我们再次创建这个文件时（文件名与之前的相同），软链接又会重新指向这个文件（inode号与之前的不同了），而硬链接不会受其影响.

当然软链接也有硬链接没有的缺点，因为链接文件包含有原文件的路径信息，所以当原文件从一个目录下移到其他目录中，再访问链接文件，系统就找不到了，

而硬链接就没有这个缺陷，你想怎么移就怎么移；还有它要系统分配额外的空间用于建立新的索引节点和保存原文件的路径。

2.java集合框架



3.java中的equals()和hashCode()

* equals()相等的两个对象他们的hashCode()肯定相等，也就是用equals()对比是绝对可靠的。
* hashCode()相等的两个对象他们的equal()不一定相等，也就是hashCode()不是绝对可靠的。

4.linux中计算权限(用数值表示)

r:4 ; w: 2; x:1

若要rwx属性则4+2+1=7；  
若要rw-属性则4+2=6；  
若要r-x属性则4+1=5。  
5. spark阶段划分：

job:以行动算子划分，一个行动算子触发一个job

stage：是job的子集，以RDD宽依赖划分，遇到shuffle做一次划分

task：是stage的子集，以并行度（分区数）划分

6．hive 中的压缩格式 RCFile、TextFile、SequenceFile各有什么区别

、

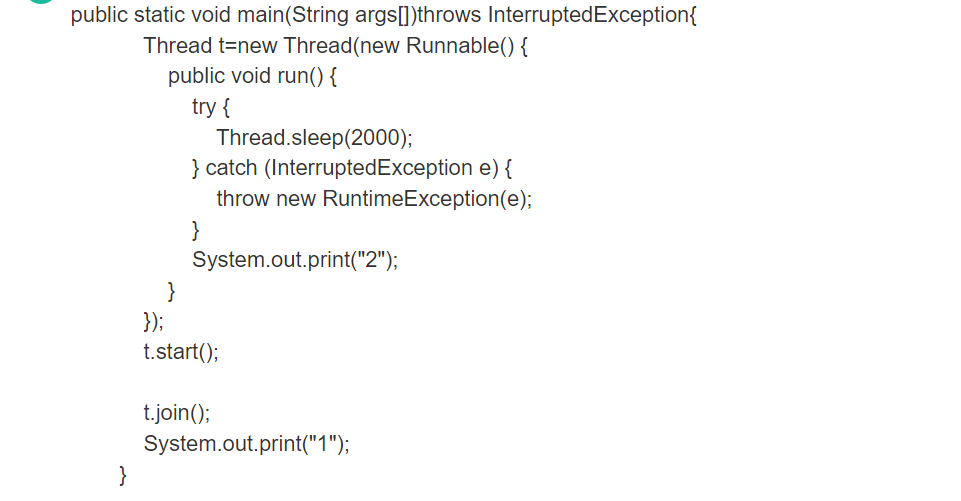
7.



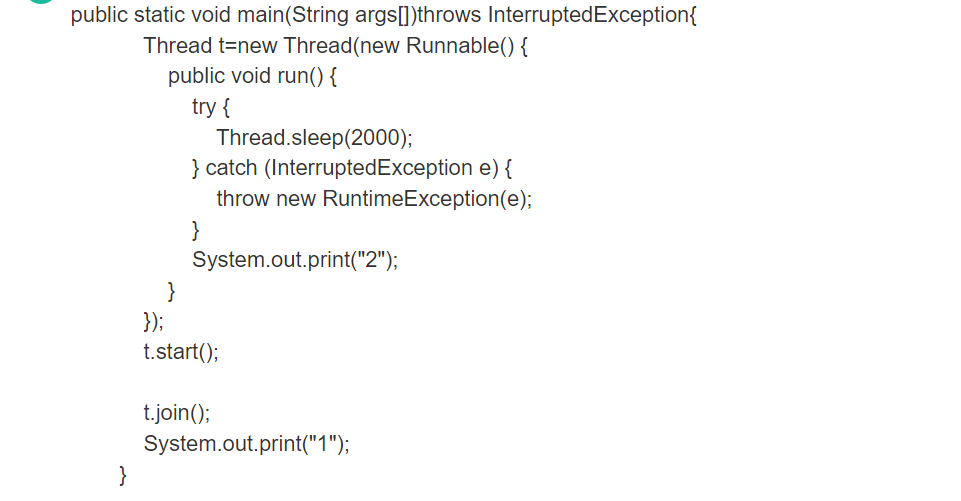
**8.java实例变量,局部变量,类变量和final变量**

* 定义在类中的变量是类的成员变量，可以不进行初始化，Java会自动进行初始化，如果是引用类型默认初始化为null,如果是基本类型例如int则会默认初始化为0。
* 局部变量是定义在方法中的变量，必须要进行初始化，否则不同通过编译。
* 被static关键字修饰的变量是静态的，静态变量随着类的加载而加载，所以也被称为类变量。
* 被final修饰发变量是常量。

**9.java多线程**

* start： 启动线程，实现了多线程运行，无需等待run方法体代码执行完毕而直接继续执行下面的代码。通过调用Thread类的start()方法来启动一个线程，这时此线程处于就绪（可运行）状态，并没有运行，一旦得到cpu时间片，就开始执行run()方法，这里方法 run()称为线程体，它包含了要执行的这个线程的内容，Run方法运行结束，此线程随即终止。
* join：很多情况下，主线程生成并起动了子线程，如果子线程里要进行大量的耗时的运算，主线程往往将于子线程之前结束，但是如果主线程处理完其他的事务后，需要用到子线程的处理结果，也就是主线程需要等待子线程执行完成之后再结束，这个时候就要用到join()方法了。所以System.out.print(21)
* 如图所示

**10.java多线程**

* start： 启动线程，实现了多线程运行，无需等待run方法体代码执行完毕而直接继续执行下面的代码。通过调用Thread类的start()方法来启动一个线程，这时此线程处于就绪（可运行）状态，并没有运行，一旦得到cpu时间片，就开始执行run()方法，这里方法 run()称为线程体，它包含了要执行的这个线程的内容，Run方法运行结束，此线程随即终止。
* join：很多情况下，主线程生成并起动了子线程，如果子线程里要进行大量的耗时的运算，主线程往往将于子线程之前结束，但是如果主线程处理完其他的事务后，需要用到子线程的处理结果，也就是主线程需要等待子线程执行完成之后再结束，这个时候就要用到join()方法了。所以System.out.print(21)
* 如图所示
* **11.**进程间5种通信方式
* 管道：速度慢，容量有限，只有父子进程能通讯；
* FIFO：任何进程间都能通讯，但速度慢；
* 消息队列：容量受到系统限制，且要注意第一次读的时候，要考虑上一次没有读完数据的问题；
* 共享内存：能够很容易控制容量，速度快，但要保持同步，比如一个进程在写的时候，另一个进程要注意读写的问题；
* 信号量：不能传递复杂消息，只能用来同步。用于实现进程间的互斥与同步，而不是用于存储进程间通信数据。

**12.四种测试方式**

* 单元测试对源程序中每一个程序单元进行测试，检查各个模块是否正确实现规定的功能，从而发现模块在编码中或算法中的错误。该阶段涉及编码和详细设计文档。
* 集成测试是基于软件需求说明书的黑盒测试，是对已经集成好的软件系统进行彻底的测试，以验证软件系统的正确性和性能等满足其规约所指定的要求，检查软件的行为和输出是否正确
* 确认测试主要是检查已实现的软件是否满足需求规格说明书中确定了的各种需求。
* 系统测试的主要目的是检查软件单位之间的接口是否正确，主要是针对程序内部结构进行测试，特别是对程序之间的接口进行测试。

**13.加密算法**

* 对称加密算法：指加密和解密使用相同密钥的加密算法。对称加密算法的优点在于加解密的高速度和使用长密钥时的难破解性。

常见对称加密算法：DES、3DES、DESX、Blowfish、IDEA、RC4、RC5、RC6和AES

* 非对称加密算法：指加密和解密使用不同密钥的加密算法，也称为公私钥加密。

例如：RSA、ECC（移动设备用）、Diffie-Hellman、El Gamal、DSA（数字签名用）

* Hash算法：Hash算法特别的地方在于它是一种单向算法，用户可以通过Hash算法对目标信息生成一段特定长度的唯一的Hash值，却不能通过这个Hash值重新获得目标信息。因此Hash算法常用在不可还原的密码存储、信息完整性校验等。

例如：MD2、MD4、MD5、HAVAL、SHA、SHA-1、HMAC、HMAC-MD5、HMAC-SHA1

**14.MYSQL最左匹配原则注意事项**

* mysql会一直向右匹配直到遇到范围查询（>、<、between、like）就停止匹配。范围列可以用到索引，但是范围列后面的列无法用到索引。
* like语句的索引问题：如果通配符 % 不出现在开头，则可以用到索引，但根据具体情况不同可能只会用其中一个前缀，在 like “value%” 可以使用索引，但是 like “%value%” 不会使用索引，走的是全表扫描
* 如果查询条件中含有函数或者表达式，将导致索引失效而进行全表查询。
* 只要列中包含有 NULL 值都将不会被包含在索引中，复合索引中只要有一列含有 NULL 值，那么这一列对于此复合索引就是无效的。所以在数据库设计时不要让字段的默认值为 NULL。