**实例分析**

在你的volumeFile文件模拟的文件系统中，依次执行下列操作：

（1）新建文件/test/Jerry；

（2）打开该文件；

（3）写入800个字节；

（4）将文件读写指针定位到第500字节；

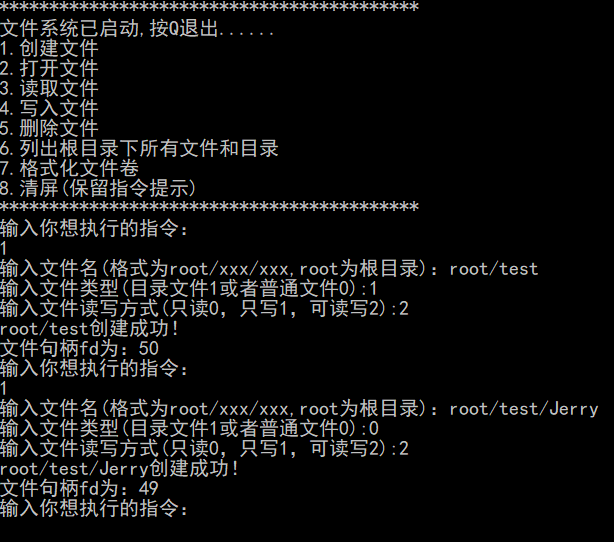
（3）读出20个字节。

观察结果是否正确，并详细解释每一步的工作过程

运行过程分析：

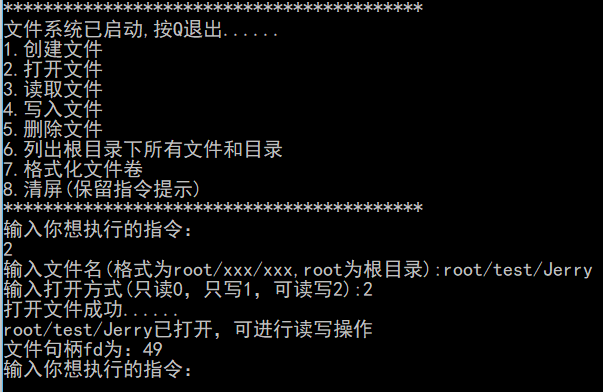
1. 新建文件/test/Jerry:

首先在根目录root下建立test目录文件，然后再在test目录下创建Jerry文件。整个过程调用两次createFile模块，创建成功后返回文件的句柄(即文件INode节点的编号)。



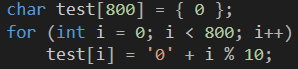
1. 打开该文件：

调用openFile模块，在openFile模块中调用SearchPath模块查找文件的INode节点，然后在用户打开文件表中查找文件是否已经打开，如果已经打开则直接返回文件句柄，如果没打开则先更新用户打开文件表，再返回文件句柄。

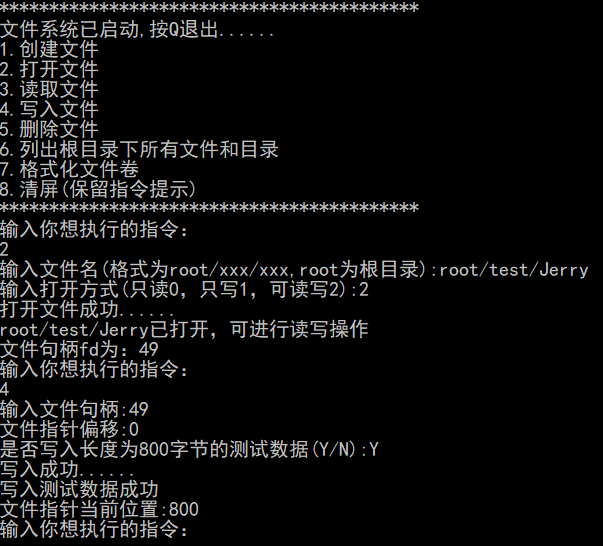


1. 写入800个字节：

测试时首先构造了一个800字节大小的数组，具体内容为0123456789的循环。构造代码如下：

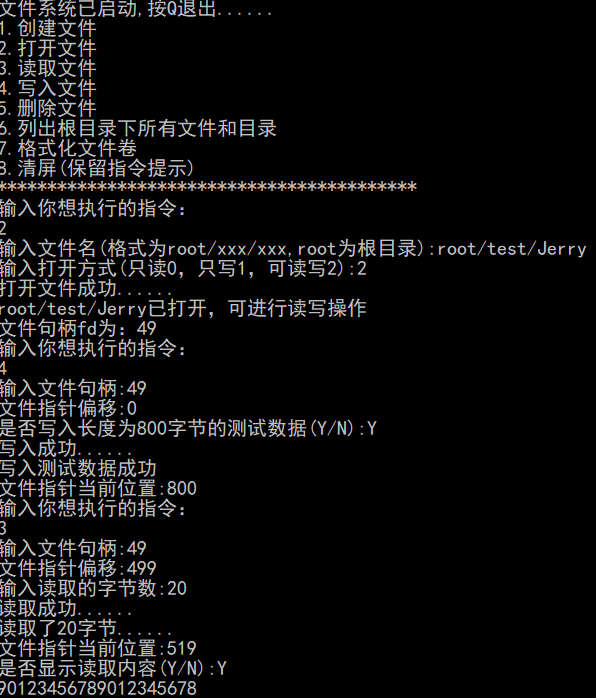


首先调用seekPos模块，定位文件读写指针位置为0，即文件开头。然后调用writeFile模块，从文件偏移地址0开始写入。由于本系统中一个物理盘块为512字节，所以在writeFile模块中，会为Jerry文件申请两个物理盘块，第一个盘块写满，第二个盘块写288字节，且文件读写指针移动到800字节处。



1. 将文件读写指针定位到第500字节，并且读20个字节：

本系统没有为用户直接提供重新定位文件读写指针的操作，而是集成在读写文件操作中的。测试时首先调用seekPos模块根据用户的输入重新定位文件读写指针，然后调用readFile模块，读取过程会跨越物理盘块，Jerry文件的第一个物理盘块仅读取12字节，第二个物理盘块读取前8个字节。根据test数组的内容来看，预期的读取出的内容应该是90123456789012345678，实际读出的内容见下图：



从文件指针偏移499(从0开始)开始读取20个字节为90123456789012345678，文件读写指针位置更新为519。由此可见，实际读取出的内容和预期读取出的内容一致，实例运行正确！