实验二：第三章 HDFS

目录

[1 实验目的 1](#_Toc42864889)

[2 实验平台 1](#_Toc42864890)

[3 实验内容和要求 1](#_Toc42864891)

[4 答案 2](#_Toc42864892)

[（1）向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，由用户指定是追加到原有文件末尾还是覆盖原有的文件； 3](#_Toc42864893)

[（2）从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名； 6](#_Toc42864894)

[（3）将HDFS中指定文件的内容输出到终端中； 7](#_Toc42864895)

[（4）显示HDFS中指定的文件的读写权限、大小、创建时间、路径等信息； 9](#_Toc42864896)

[（5）给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，如果该文件是目录，则递归输出该目录下所有文件相关信息； 10](#_Toc42864897)

[（6）提供一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录； 12](#_Toc42864898)

[（7）提供一个HDFS的目录的路径，对该目录进行创建和删除操作。创建目录时，如果目录文件所在目录不存在则自动创建相应目录；删除目录时，由用户指定当该目录不为空时是否还删除该目录； 14](#_Toc42864899)

[（8）向HDFS中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾； 17](#_Toc42864900)

[（9） 删除HDFS中指定的文件； 20](#_Toc42864901)

[（10） 在HDFS中，讲文件从源路径移动到目的路径。 21](#_Toc42864902)

# 实验目的

1. 理解HDFS在Hadoop体系结构中的角色；

2. 熟练使用HDFS操作常用的Shell命令；

3. 熟悉HDFS操作常用的Java API。

# 实验平台

操作系统：Linux

Hadoop版本：2.6.0或以上版本

JDK版本：1.8

Java IDE：Idea

# 实验内容和要求

1. 编程实现以下指定功能，并利用Hadoop提供的Shell命令完成相同任务：
2. 向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，由用户指定是追加到原有文件末尾还是覆盖原有的文件；
3. 从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名；
4. 将HDFS中指定文件的内容输出到终端中；
5. 显示HDFS中指定的文件的读写权限、大小、创建时间、路径等信息；
6. 给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，如果该文件是目录，则递归输出该目录下所有文件相关信息；
7. 提供一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录；
8. 提供一个HDFS的目录的路径，对该目录进行创建和删除操作。创建目录时，如果目录文件所在目录不存在则自动创建相应目录；删除目录时，由用户指定当该目录不为空时是否还删除该目录；
9. 向HDFS中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾；
10. 删除HDFS中指定的文件；
11. 在HDFS中，讲文件从源路径移动到目的路径。

# 答案

## （1）向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，由用户指定是追加到原有文件末尾还是覆盖原有的文件；

**package** com.hwua.com;  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FSDataOutputStream;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
**import** java.io.FileInputStream;  
**import** java.io.IOException;  
*/\*  
\* 第一题：向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，则由用户来指定是追加到原有文件末尾  
\* 还是覆盖原有的文件。  
\* shell：  
\* $ hadoop fs -test -e /test/1.txt  
\* $ echo $?  
\* $hadoop fs -appendToFile local.txt /test/1.txt #追加到原文件末尾  
\* $hadoop fs -put -f local.txt /test/1.txt #覆盖原来的文件  
\* $ if  
\* $ hadoop fs -test -e /test/1.txt  
\* $ then  
\* $ hadoop fs -appendToFile local.txt /test/1.txt  
\* $ else  
\* $ hadoop fs -put -f local.txt /test/1.txt  
\* $ fi  
\* \*/***public class** Demo1 {  
 */\*写了一个静态方法，通过类名·出来  
 \*如果不是，要实例化才能使用，  
 \* 什么是实例化？就是创建一个对象Configuration conf = new Configuration();  
 \* new后面写什么该类的构造器  
 \*方法名为类名，而且方法没有返回值。并不是返回值为空  
 \*这个类中有几个构造器，那么就有几种方式去创建该类的对象  
 \*/  
 // 判断路径是否存在* **public static boolean** test(Configuration conf,String path) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 **return** fs.exists(**new** Path(path));  
 }  
 */\*  
 \* 复制文件到指定路径  
 \* 若路径已存在，则进行覆盖  
 \* \*/* **public static void** copyFromLocalFile(Configuration conf,String localFilePath,  
 String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path localPath = **new** Path(localFilePath);  
 Path remotePath =**new** Path(remoteFilePath);  
 */\*fs.copyFromLocalFile  
 第一个参数表示是否删除源文件，  
 第二个参数表示是否覆盖\*/* fs.copyFromLocalFile(**false**,**true**,localPath,remotePath);  
 fs.close();  
 }  
 *//追加文件内容* **public static void** appendToFile(Configuration conf,String localFilePath,  
 String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath =**new** Path(remoteFilePath);  
 *//创建一个文件读入流* FileInputStream in = **new** FileInputStream(localFilePath);  
 *//创建一个文件输出流，输出的内容将追加到文件末尾* FSDataOutputStream out =fs.append(remotePath);  
 *//读写文件内容  
 /\*声明了一个字节数组  
 \*一次读1024字节  
 \* ASCII码 American Standard Code for Information Interchang  
 \* read=-1 表示没有读到  
 \*/* **byte**[] data = **new byte**[1024];  
 **int** read = -1;  
 **while** ((read=in.read(data))>0){  
 *//表示从0开始读，一直读到read* out.write(data,0,read);  
 }  
 out.close();  
 in.close();  
 fs.close();  
 }  
 **public static void** main(String[] args) **throws** IOException {  
 */\*该类的构造器  
 \*方法名为类名，而且方法没有返回值。并不是返回值为空  
 \*这个类中有几个构造器，那么就有几种方式去创建该类的对象  
 \*/* Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 *//本地路径* String localFilePath =**"D:\\Java\\test\\123.txt"**;  
 *//HDFS路径* String remoteFilePath=**"/test/1.txt"**;  
 *//若文件存在，则追加到文件末尾* String choice =**"append"**;  
 *//若文件存在则覆盖  
 //String choice ="overwrite";  
 //判断文件是否存在* Boolean fileExists =**false**;  
 **if** (Demo1.*test*(conf,remoteFilePath)){  
 fileExists = **true** ;  
 System.***out***.println(remoteFilePath+**"已存在."**);  
 }**else** {  
 System.***out***.println(remoteFilePath+**"不存在."**);  
 }  
 *//进行处理* **if** (!fileExists) {*//如果文件不存在，则进行上传* Demo1.*copyFromLocalFile*(conf,localFilePath,remoteFilePath);  
 System.***out***.println(localFilePath+**"已上传至"**+remoteFilePath);  
 } **else if** (choice.equals(**"overwrite"**)) {  
 Demo1.*copyFromLocalFile*(conf,localFilePath,remoteFilePath);  
 System.***out***.println(localFilePath+**"已覆盖"**+remoteFilePath);  
 }**else if** (choice.equals(**"append"**)){  
 Demo1.*appendToFile*(conf,localFilePath,remoteFilePath);  
 System.***out***.println(localFilePath+**"已追加至"**+remoteFilePath);  
 }  
 }  
}

（2）从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
**import** java.io.File;  
**import** java.io.IOException;  
  
*/\*  
\* 从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，  
\* 则自动对下载的文件重命令  
\*  
\* $ if  
\* $ hadoop fs -test -e file:////home/master/apps/a/test/1.txt  
\* $ then  
\* $ hadoop fs -get /test/1.txt ./2.txt  
\* $ else  
\* $ hadoop fs -get /test/1.txt ./1.txt  
\* $ fi  
\* \*/***public class** Demo2 {  
*/\*  
\* 下载文件到本地  
\*判断本地路径是否已存在，若已存在，则自动进行重命名  
\* \*/***public static void** copyToLocal(Configuration conf,String remoteFilePath,String localFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 File f = **new** File(localFilePath);  
 */\*  
 \* 如果文件名存在，自动重命名(在文件名后面加上，\_0,\_1...)  
 \* \*/* **if** (f.exists()){  
 System.***out***.println(localFilePath+**"已存在"**);  
 *//后面加的名字* Integer i = 1;  
 **while** (**true**){  
 f = **new** File(localFilePath+**"\_"**+i.toString());  
 **if** (!f.exists()){  
 localFilePath = localFilePath+**"\_"**+i.toString();  
 **break**;  
 }  
 }  
 System.***out***.println(**"将重命名为"**+localFilePath);  
 }  
 *//下载文件到本地* Path localPath = **new** Path(localFilePath);  
 fs.copyToLocalFile(remotePath,localPath);  
 fs.close();  
}  
  
 **public static void** main(String[] args) {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 *//本地路径* String localFilePath =**"D:\\Java\\test1\\1.txt"**;  
 *//HDFS路径* String remoteFilePath=**"/test/1.txt"**;  
  
  
 **try** {  
 Demo2.*copyToLocal*(conf,remoteFilePath,localFilePath);  
 System.***out***.println(**"下载完成"**);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
}

## （3）将HDFS中指定文件的内容输出到终端中；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FSDataInputStream;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
  
*/\*  
\*  
\* 将HDFS中指定文件的内容输出到终端中  
\* $ hadoop fs -cat /test/1.txt  
\* \*/***public class** Demo3 {  
 */\*  
 \* 读取文件内容  
 \* \*/* **public static void** cat(Configuration conf,String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 FSDataInputStream in = fs.open(remotePath);  
 BufferedReader d = **new** BufferedReader(**new** InputStreamReader(in));  
 String line = **null** ;  
 **while** ((line = d.readLine())!=**null**){  
 System.***out***.println(line);  
 }  
 d.close();  
 in.close();  
 fs.close();  
 }  
  
 **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 *//HDFS路径* String remoteFilePath=**"/test/3.txt"**;  
  
 System.***out***.println(**"读取文件"**+remoteFilePath);  
 Demo3.*cat*(conf,remoteFilePath);  
 System.***out***.println(**"\n 读取完成"**);  
 }  
}

## （4）显示HDFS中指定的文件的读写权限、大小、创建时间、路径等信息；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FileStatus;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
  
**import** java.io.IOException;  
**import** java.text.SimpleDateFormat;  
  
*/\*  
\* 显示HDFS 中指定的文件的读写权限、大小、创建时间、路径等信息  
\* $ hadoop fs -ls -h /test/1.txt  
\*  
\* \*/***public class** Demo4 {  
 */\*  
 \* 显示指定文件的信息  
 \* \*/* **public static void** ls(Configuration conf,String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 *//查看目录* FileStatus[] fileStatuses = fs.listStatus(remotePath);  
 **for** (FileStatus s: fileStatuses  
 ) {  
 */\* .toString() 转换成字符串  
 \* \*/* System.***out***.println(**"路径："**+s.getPath().toString());  
 System.***out***.println(**"权限："**+s.getPermission().toString());  
 System.***out***.println(**"大小："**+s.getLen());  
 */\*  
 \* 返回 的是时间戳，转化为时间日期格式  
 \* \*/* Long timeStamp = s.getModificationTime();  
 SimpleDateFormat format = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 String date = format.format(timeStamp);  
 System.***out***.println(**"时间："**+date);  
 }  
 fs.close();  
 }  
  
 */\*  
 \* 主函数  
 \* \*/* **public static void** main(String[] args) {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 *//HDFS路径* String remoteFilePath=**"/test/3.txt"**;  
  
  
 System.***out***.println(**"读取文件信息："**+remoteFilePath);  
 **try** {  
 Demo4.*ls*(conf,remoteFilePath);  
 System.***out***.println(**"\n 读取完成"**);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

## （5）给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，如果该文件是目录，则递归输出该目录下所有文件相关信息；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.\*;  
  
**import** java.io.IOException;  
**import** java.text.SimpleDateFormat;  
  
*/\*  
\* 给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、  
\* 路径等信息，如果改文件是目录，则递归输出该目录下所有文件相关信息*

*\* hadoop fs -ls -R -h /test  
\* \*/***public class** Demo5 {  
  
 */\*  
 \* 显示指定文件夹下所有文件的信息（递归）  
 \* \*/* **public static void** lsDir(Configuration conf,String remoteDir) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path dirPath = **new** Path(remoteDir);  
  
 */\*  
 \* 递归获取目录下的所有文件  
 \* \*/* RemoteIterator<LocatedFileStatus> remoteIterator =fs.listFiles(dirPath,**true**);  
 */\* 输出每个文件的信息\*/* **while** (remoteIterator.hasNext()){  
 FileStatus s = remoteIterator.next();  
 System.***out***.println(**"路径："** + s.getPath().toString());  
 System.***out***.println(**"权限："** + s.getPermission().toString());  
 System.***out***.println(**"大小："** + s.getLen());  
 */\* 返回的是时间戳，转化为时间日期格式 \*/* Long timeStamp = s.getModificationTime();  
 SimpleDateFormat format = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 String date = format.format(timeStamp);  
 System.***out***.println(**"时间："** +date);  
 }  
 fs.close();  
 }  
  
 */\*  
 \* 主函数  
 \* \*/* **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 *//HDFS路径 的目录路径* String remoteFilePath=**"/test"**;  
  
  
 System.***out***.println(**"读取文件信息："** + remoteFilePath);  
 Demo4.*ls*(conf,remoteFilePath);  
 System.***out***.println(**"\n 读取完成"**);  
  
 }  
}

## （6）提供一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FSDataOutputStream;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
  
**import** java.io.IOException;  
  
*/\*  
\* 创建一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，  
\* 则自动创建目录  
\*  
\* shell ：  
\* $ if  
\* $ hadoop fs -test -d /test/demo6  
\* $ then  
\* $ hadoop fs -touchz /test/demo6/6.txt  
\* $ else  
\* $ hadoo fs -mkdir -p /test/demo6 && hadoop fs -touchz /test/demo6/6.txt  
\* $ fi  
\* $ hadoop fs -rm /test/demo6/6.txt  
\* \*/***public class** Demo6 {  
 */\*  
 \* 判断路径是否存在  
 \* \*/* **public static boolean** test(Configuration conf,String path) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 **return** fs.exists(**new** Path(path));  
 }  
  
 */\*  
 \* 创建目录  
 \* \*/* **public static boolean** mkdir(Configuration conf,String remoteDir) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path dirPath = **new** Path(remoteDir);  
 **boolean** result = fs.mkdirs(dirPath);  
 fs.close();  
 **return** result;  
 }  
 */\*  
 \* 创建文件  
 \* \*/* **public static void** touchz(Configuration conf,String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 FSDataOutputStream outputStream = fs.create(remotePath);  
 outputStream.close();  
 fs.close();  
 }  
  
 */\*  
 \* 删除文件  
 \* \*/* **public static boolean** rm(Configuration conf,String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 **boolean** result = fs.delete(remotePath,**false**);  
 fs.close();  
 **return** result;  
  
 }  
  
 */\*  
 \* 主函数  
 \* \*/* **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 *//指定用户为root用户* System.*setProperty*(**"HADOOP\_USER\_NAME"**,**"root"**);  
 *//HDFS路径的目录路径* String remoteFilePath=**"/test/demo6/demo6.txt"**;  
 *//HDFS路径对应的目录* String remoteDir =**"/test/demo6"**;  
  
  
 */\* 判断路径是否存在，存在则删除，否则进行创建\*/* **if** (Demo6.*test*(conf,remoteFilePath)){  
 *//进行删除* Demo6.*rm*(conf,remoteFilePath);  
 System.***out***.println(**"删除路径："**+remoteFilePath);  
 }**else**{  
 **if** (!Demo6.*test*(conf,remoteDir)){*//若目录不存在，则进行创建* Demo6.*mkdir*(conf,remoteDir);  
 System.***out***.println(**"创建文件夹："**+ remoteDir);  
 }  
 Demo6.*touchz*(conf,remoteFilePath);  
 System.***out***.println(**"创建路径："**+ remoteFilePath);  
 }  
 }  
  
  
}

## （7）提供一个HDFS的目录的路径，对该目录进行创建和删除操作。创建目录时，如果目录文件所在目录不存在则自动创建相应目录；删除目录时，由用户指定当该目录不为空时是否还删除该目录；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.LocatedFileStatus;  
**import** org.apache.hadoop.fs.Path;  
**import** org.apache.hadoop.fs.RemoteIterator;  
  
**import** java.io.IOException;  
  
*/\*  
\* 提供一个HFDS的目录的路径，对该目录进行创建和删除操作。  
\* 创建目录时，如果目录文件所在目录不存在，则自动创建相应目录；  
\* 删除目录时，由用户指定当该目录不为空时，是否还删除该目录  
\*  
\* shell操作：  
\* $ hadoop fs -mkdir -p /test1/test2  
\* 删除目录的命令如下：  
\* $ hadoop fs -rmdir /test1/test2  
\* 执行完之后，如果目录非空，会有提示。如果要强制删除，可以使用以下命令：  
\* $ hadoop fs -rm -R /test1/test2  
\*  
\* \*/***public class** Demo7 {  
 */\*  
 \* 判断路径是否存在  
 \* \*/* **public static boolean** test(Configuration conf,String path) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 **return** fs.exists(**new** Path(path));  
 }  
  
 */\*  
 \* 判断目录是否为空  
 \* true: 空；false 非空  
 \* \*/* **public static boolean** isDirEmpty(Configuration conf,String remoteDir) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path dirPath = **new** Path(remoteDir);  
 RemoteIterator<LocatedFileStatus> remoteIterator = fs.listFiles(dirPath,**true**);  
 **return** !remoteIterator.hasNext();  
 }  
  
 */\*  
 \* 创建目录  
 \* \*/* **public static boolean** mkdir(Configuration conf,String remoteDir) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path dirPath = **new** Path(remoteDir);  
 **boolean** result = fs.mkdirs(dirPath);  
 fs.close();  
 **return** result;  
 }  
  
 */\*  
 \* 删除目录  
 \* \*/* **public static boolean** rmDir(Configuration conf,String remoteDir) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path dirPath = **new** Path(remoteDir);  
 *//第二个参数表示是否诋毁删除所有文件* **boolean** result = fs.delete(dirPath,**true**);  
 fs.close();  
 **return** result;  
 }  
 */\* 主函数 \*/* **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 System.*setProperty*(**"HADOOP\_USER\_NAME"**,**"root"**);  
 *//HDFS路径对应的目录* String remoteDir =**"/test/test1"**;  
 *// 是否强制删除* Boolean forceDelete = **false** ;  
  
 **if** (!Demo7.*test*(conf,remoteDir)){  
 *//创建目录* Demo7.*mkdir*(conf,remoteDir);  
 System.***out***.println(**"创建目录："** + remoteDir);  
 }**else** {  
 *//目录为空或者强制删除* **if** (Demo7.*isDirEmpty*(conf,remoteDir) || forceDelete){  
 Demo7.*rmDir*(conf,remoteDir);  
 System.***out***.println(**"删除目录："** + remoteDir);  
 }**else** { *//目录不为空* System.***out***.println(**"目录不为空，不删除："** + remoteDir);  
 }  
 }  
 }  
}

## （8）向HDFS中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾；

**package** com.hwua.com;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FSDataOutputStream;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
  
**import** java.io.FileInputStream;  
**import** java.io.IOException;  
  
*/\*  
\* 向HDFS中指定的文件追加内容，由用户指定内容追加到原有的文件开头或者结尾。  
\* shell:  
\* $ hadoop fs -appendToFile local.txt /test/test.txt  
\* \*/***public class** Demo8 {  
 */\*  
 \* 判断路径是否存在  
 \* \*/* **public static boolean** test(Configuration conf,String path) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 **return** fs.exists(**new** Path(path));  
 }  
  
 */\*  
 \* 追加文本内容  
 \* \*/* **public static void** appendContentToFile(Configuration conf,String content,  
 String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 */\* 创建一个文件输出流，输出的内容将追加到文件末尾\*/* FSDataOutputStream out = fs.append(remotePath);  
 out.write(content.getBytes());  
 out.close();  
 fs.close();  
 }  
  
 */\*  
 \* 追加文件内容  
 \* \*/* **public static void** appendToFile(Configuration conf,String localFilePath,  
 String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 */\*创建一个文件读入流\*/* FileInputStream in = **new** FileInputStream(localFilePath);  
 */\*创建一个文件输出流，输出的内容将追加到文件末尾\*/* FSDataOutputStream out = fs.append(remotePath);  
 */\*读写文件内容\*/* **byte**[] data = **new byte**[1024];  
 **int** read = -1;  
 **while** ((read = in.read(data))>0){  
 out.write(data,0,read);  
 }  
 out.close();  
 in.close();  
 fs.close();  
 }  
  
 */\*  
 \* 移动文件到本地  
 \* 移动后删除源文件  
 \* \*/* **public static void** moveToLocalFile(Configuration conf,String remoteFilePath  
 ,String localFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 Path localPath = **new** Path(localFilePath);  
 fs.moveToLocalFile(remotePath,localPath);  
 }  
  
 */\*  
 \* 创建文件  
 \* \*/* **public static void** touchz(Configuration conf,String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = **new** Path(remoteFilePath);  
 FSDataOutputStream outputStream = fs.create(remotePath);  
 outputStream.close();  
 fs.close();  
 }  
  
  
  
 */\*  
 \* 主函数  
 \* \*/* **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 System.*setProperty*(**"HADOOP\_USER\_NAME"**,**"root"**);  
 *//HDFS路径对应的目录* String remoteFilePath =**"/test/test1"**;  
 String content = **"新追加的内容 \n"**;  
 String choice = **"after"**; *//追加到文件末尾  
 // String choice = "before"; //追加到文件开头  
  
  
 /\*判断文件是否存在\*/* **if** (!Demo8.*test*(conf,remoteFilePath)){  
 System.***out***.println(**"文件不存在："**+ remoteFilePath);  
 }**else** {  
 **if** (choice.equals(**"after"**)){*//追加在文件末尾* Demo8.*appendContentToFile*(conf,content,remoteFilePath);  
 System.***out***.println(**"已追加内容到文件末尾"**+ remoteFilePath);  
 } **else if** (choice.equals(**"before"**)) {*//追加在文件开头  
 /\*没有相应的API可以直接操作，因此先把文件移动到本地\*/  
 /\*创建一个新的HDFS,再按顺序追加内容\*/* String localTmpPath=**"/test/test1/tmp.txt"**;  
 *//移动到本地* Demo8.*moveToLocalFile*(conf,remoteFilePath,localTmpPath);  
 *//创建一个新文件* Demo8.*touchz*(conf,remoteFilePath);  
 *//先写入新内容* Demo8.*appendContentToFile*(conf,content,remoteFilePath);  
 *//再写入原来的内容* Demo8.*appendToFile*(conf,localTmpPath,remoteFilePath);  
 System.***out***.println(**"已追加内容导文件开头："**+remoteFilePath);  
 }  
 }  
 }  
}

1. 删除HDFS中指定的文件；

**package** com.jsck.hdfs.javaapi;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
  
**import** java.io.IOException;  
  
*/\*  
\* （9） 删除HDFS中指定的文件；  
\* shell:  
\*  
\* hadoop fs -rm 1.txt  
\* \*/***public class** HdfsDemo9 {  
 */\*删除文件\*/* **public static boolean** rm(Configuration conf,String remoteFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath =**new** Path(remoteFilePath);  
 *//删除 ，第二个空是：是否递归删除* **boolean** result = fs.delete(remotePath,**false**);  
 fs.close();  
 **return** result;  
 }  
 **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 conf.set(**"fs.hdfs.impl"**,**"org.apache.hadoop.hdfs.DistributedFileSystem"**);  
 System.*setProperty*(**"HADOOP\_USER\_NAME"**,**"root"**);  
 *// HDFS路径* String remoteFilePath = **"/test2/3.txt"**;  
 **if** (HdfsDemo9.*rm*(conf,remoteFilePath)){  
 System.***out***.println(**"文件已删除"**+remoteFilePath);  
 }**else** {  
 System.***out***.println(**"删除失败"**);  
 }  
 }  
}

1. 在HDFS中，讲文件从源路径移动到目的路径。

**package** com.jsck.hdfs.javaapi;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.FileSystem;  
**import** org.apache.hadoop.fs.Path;  
  
**import** java.io.IOException;  
  
*/\*  
\* （10） 在HDFS中，讲文件从源路径移动到目的路径。  
\* shell  
\* hadoop fs -mv 1.txt 2.txt  
\* \*/***public class** HdfsDemo10 {  
 */\*  
 \* 移动文件  
 \* \*/* **public static boolean** mv(Configuration conf,String remoteFilePath  
 ,String remoteToFilePath) **throws** IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path srcPath = **new** Path(remoteFilePath);  
 Path dstPath = **new** Path(remoteToFilePath);  
 *//移动（重命名）* **boolean** result = fs.rename(srcPath,dstPath);  
 fs.close();  
 **return** result;  
 }  
 */\*  
 \* 主函数  
 \* \*/* **public static void** main(String[] args) **throws** IOException {  
 Configuration conf = **new** Configuration();  
 conf.set(**"fs.defaultFS"**,**"hdfs://master:9000"**);  
 conf.set(**"fs.hdfs.impl"**,**"org.apache.hadoop.hdfs.DistributedFileSystem"**);  
 System.*setProperty*(**"HADOOP\_USER\_NAME"**,**"root"**);  
 *// 原来的路径* String remoteFilePath = **"/test2/3.txt"**;  
 *// 移动的目标路径* String remoteToFilePath = **"/test1/2.txt"**;  
  
 **if** (HdfsDemo10.*mv*(conf,remoteFilePath,remoteToFilePath)){  
 System.***out***.println(**"将文件"** + remoteFilePath +**"移动到"** + remoteToFilePath);  
 }**else** {  
 System.***out***.println(**"操作失败（文件不存在或者移动失败）"**);  
 }  
 }  
}