## 实验二：熟悉常用的HDFS操作

### 一、实验目的

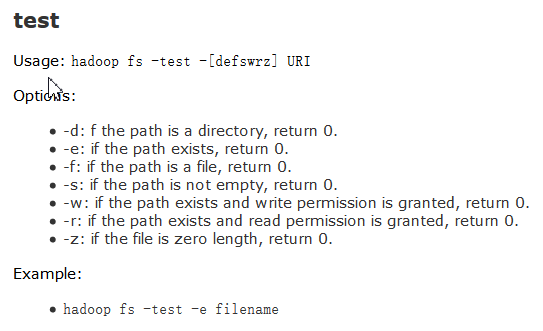
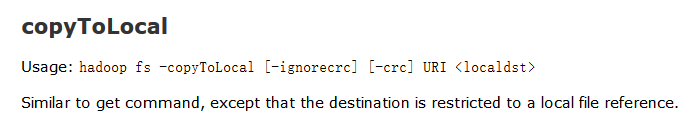
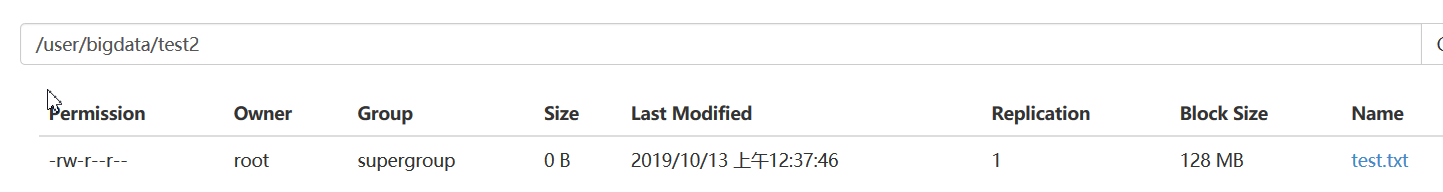
* 理解HDFS在Hadoop体系结构中的角色，
* 熟练使用HDFS操作常用的Shell命令，
* 熟悉HDFS操作常用的Java API。

### 二、实验平台

* 操作系统：Linux（建议Ubuntu18.04）
* JDK版本：open-jdk-1.11
* Hadoop版本：3.2.1
* IDE：eclipse-4.3

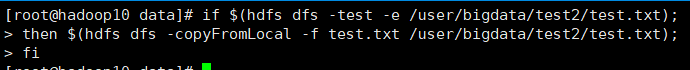
### 三、实验步骤

（一）编程实现以下功能，并利用Hadoop提供的Shell命令完成相同任务：

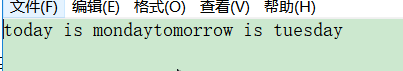
1. 向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，则由用户来指定是追加到原有文件末尾还是覆盖原有的文件；
   1. Shell命令
      1. 用到的HDFS操作
         1. 
         2. 
      2. 向HDFS中上传的目录选择是/user/bigdata/test2
      3. 查看HDFS中该目录下的内容
         1. 
         2. 可以将该文件下载下来,查看文件内容为
            1. 
         3. 存在一个名为text.txt的文件
      4. 使用shell命令,从Linux本地向HDFS上传同名文件text.txt
         1. 操作--覆盖操作
            1. 本地的/usr/local/bigdataTest/data目录下test.txt文件内容如下



* + - * 1. 执行shell命令,功能是通过使用if语句来判断HDFS上是否存在同名文件,如果存在,则将原有文件覆盖,



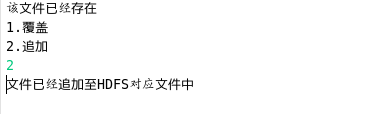
* + - * 1. 再次进入HDFS的web端,查看test.txt文件是否覆盖成了新的内容

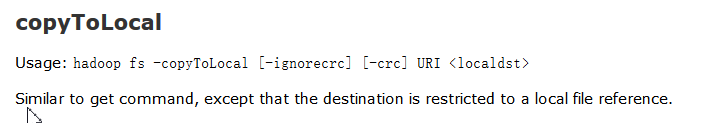
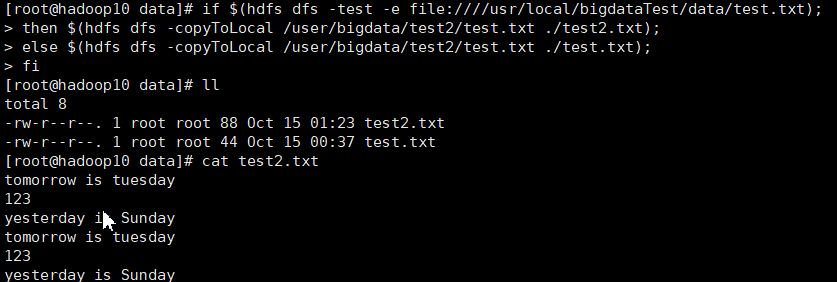


发现内容已经进行了覆盖

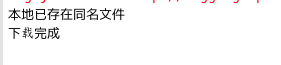
* 1. java代码

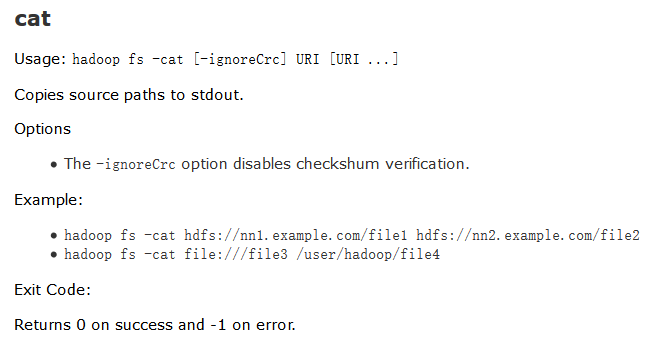
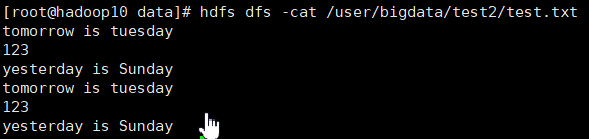
|  |
| --- |
| package com.fc.two;  import java.io.FileInputStream;  import java.io.IOException;  import java.util.Scanner;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FSDataOutputStream;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  import sun.tools.tree.LengthExpression;  public class TestTwo\_1 {    /\*\*  \*判断HDFS上文件是否存在  \* @param args  \*/  public static boolean isExists(Configuration conf,String path) throws Exception{  FileSystem fs = FileSystem.get(conf);  return fs.exists(new Path(path));  }    /\*\*  \* 上传本地文件至HDFS，如果已经存在，则进行覆盖  \* @param args  \*/  public static void overwrite(Configuration conf,String localPath,String remotePath)throws Exception{  FileSystem fs = FileSystem.get(conf);  fs.copyFromLocalFile(false, true, new Path(localPath), new Path(remotePath));  //关闭数据流  fs.close();  }    /\*\*  \* 上传本地文件到HDFS，如果已经存在，则进行末尾追加  \* @param args  \*/  public static void append(Configuration conf, String localPath,  String remotePath) throws Exception{  FileSystem fs =null;  //文件输入流  FileInputStream in =null;  FSDataOutputStream out =null;  try {  fs = FileSystem.get(conf);  in = new FileInputStream(localPath);  out = fs.append(new Path(remotePath));  //以字节数组的形式进行读取  byte[] data = new byte[1024];  int length = -1;  while ((length=in.read(data)) > 0) {  out.write(data, 0, length);  }  } catch (Exception e) {  e.printStackTrace();  }finally{  out.close();  in.close();  fs.close();  }  }    public static void main(String[] args) throws Exception {  //创建配置文件  Configuration conf = new Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  //如集群节点少于3个在运行时就会抛异常，解决方案，修改以下两个配置  conf.set("dfs.client.block.write.replace-datanode-on-failure.policy","NEVER");  conf.set("dfs.client.block.write.replace-datanode-on-failure.enable","true");  //上传的本地路径  String localPath = "/usr/local/bigdataTest/data/test.txt";  //上传到HDFS的路径  String remotePath = "/user/bigdata/test2/test.txt"; // HDFS路径  //查看HDFS上该文件是否已经存在，并打印信息反馈给用户  System.out.println(isExists(conf, remotePath)?"该文件已经存在":"该文件不存在");  //模拟用户选择  Scanner scanner = new Scanner(System.in);  System.out.println("1.覆盖");  System.out.println("2.追加");  int choose = scanner.nextInt();  switch (choose) { //1代表append 2代表overwrite  case 1:  overwrite(conf, localPath, remotePath);  System.out.println("文件已经覆盖至HDFS对应文件中");  break;  case 2:  append(conf, localPath, remotePath);  System.out.println("文件已经追加至HDFS对应文件中");  break;  }  }    } |



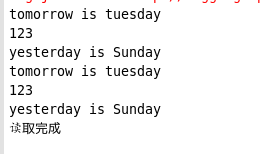
1. 从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名；
   1. shell命令
      1. 用到的HDFS操作(前面重复出现的不再赘述)
         1. 
      2. 
      3. if语句用来判断本地目录中是否存在test.txt文件,如果存在就下载并改名为test2.txt,否则下载为test.txt
   2. java代码

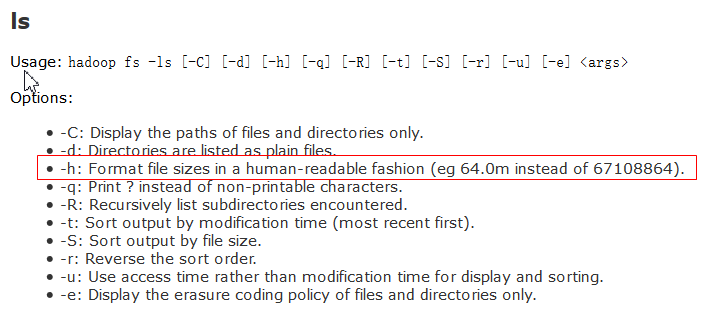
|  |
| --- |
| **package** com.fc.two;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.\*;  **import** java.io.\*;  **import** java.util.Arrays;  **public** **class** TestTwo\_2 {  /\*\*  \* 从HDFS下载文件到本地  \* 1.检查本次下载的目录中是否存在同名文件，存在则改名，不存在则原名下载  \*/  **public** **static** **void** copyToLocal(Configuration conf, String remotePath,  String localPath) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  File file = **new** File(localPath);  //如果本地存在同名文件，自动改名  **if** (file.exists()) {  System.***out***.println("本地已存在同名文件");  //切割字符串，从.前面添加后缀  String[] arr = localPath.split("\\.");  localPath = arr[0]+"2."+arr[1];  file = **new** File(localPath);  }  // 下载文件到本地  fs.copyToLocalFile(**new** Path(remotePath), **new** Path(localPath));  fs.close();  }  **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  //设置路径  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  //本地Linux路径  String localPath = "/usr/local/bigdataTest/data/test.txt";  //目标下载文件路径  String remotePath = "/user/bigdata/test2/test.txt";  **try** {  *copyToLocal*(conf, remotePath, localPath);  System.***out***.println("下载完成");  } **catch** (Exception e) {  e.printStackTrace();  }  }  } |



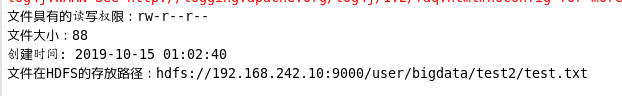
1. 将HDFS中指定文件的内容输出到终端中；
   1. shell命令
      1. 用到的HDFS命令
         1. 
      2. 
   2. java代码

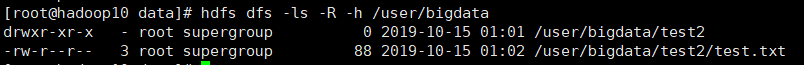
|  |
| --- |
| **package** com.fc.two;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.FSDataInputStream;  **import** org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.Path;  **public** **class** TestTwo\_3 {    **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  //设置HDFS文件路径  String remotePath = "/user/bigdata/test2/test.txt";  **try** {  *cat*(conf,remotePath);  System.***out***.println("读取完成");  } **catch** (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* 将HDFS中指定文件的内容输出到终端中  \* **@throws** IOException  \*/  **public** **static** **void** cat(Configuration conf, String remotePath) **throws** IOException{  FileSystem fs=**null**;  FSDataInputStream in=**null**;  BufferedReader read=**null**;  **try** {  fs = FileSystem.*get*(conf);  //创建文件输入输出流  in = fs.open(**new** Path(remotePath));  read = **new** BufferedReader(**new** InputStreamReader(in));  String line = **null**;  **while** ((line = read.readLine()) != **null**) {//读取到末尾时停止读取  System.***out***.println(line);  }  } **catch** (Exception e) {  e.printStackTrace();  }**finally**{  read.close();  in.close();  fs.close();  }  }  } |



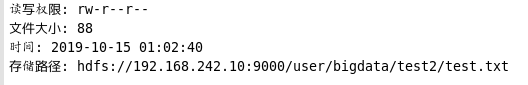
1. 显示HDFS中指定的文件的读写权限、大小、创建时间、路径等信息；
   1. shell命令
      1. 用到的HDFS命令
         1. 
      2. 
   2. java代码

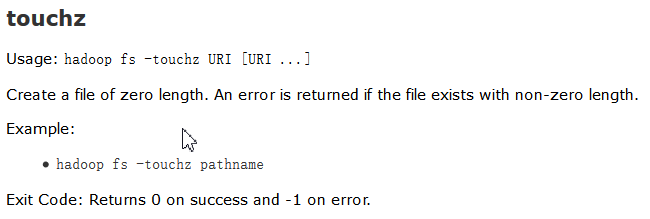
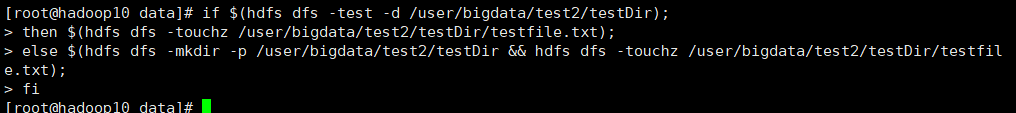
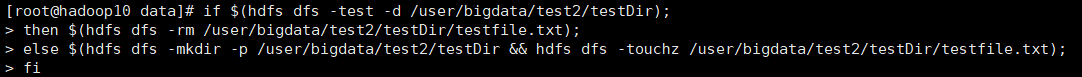
|  |
| --- |
| **package** com.fc.two;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.\*;  **import** java.io.\*;  **import** java.text.SimpleDateFormat;  **public** **class** TestTwo\_4 {  /\*\*  \* ls命令查看指定文件的权限  \*/  **public** **static** **void** ls(Configuration conf, String remoteFilePath)  **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  //获取文件数组  FileStatus[] status = fs.listStatus(**new** Path(remoteFilePath));  **for** (FileStatus s : status) {  System.***out***.println("文件具有的读写权限：" + s.getPermission().toString());  System.***out***.println("文件大小："+s.getLen());  /\*\*  \* 由于存储的是时间戳，因此要进行日期格式的转换  \*/  Long timeStamp = s.getModificationTime();  SimpleDateFormat format = **new** SimpleDateFormat(  "yyyy-MM-dd HH:mm:ss");  String date = format.format(timeStamp);  System.***out***.println("创建时间: " + date);  System.***out***.println("文件在HDFS的存放路径：" + s.getPath().toString());  }  fs.close();  }  **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  String remoteFilePath = "/user/bigdata/test2/test.txt";  **try** {  *ls*(conf, remoteFilePath);  } **catch** (Exception e) {  e.printStackTrace();  }  }  } |



1. 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；
   1. shell命令
      1. 
   2. java代码

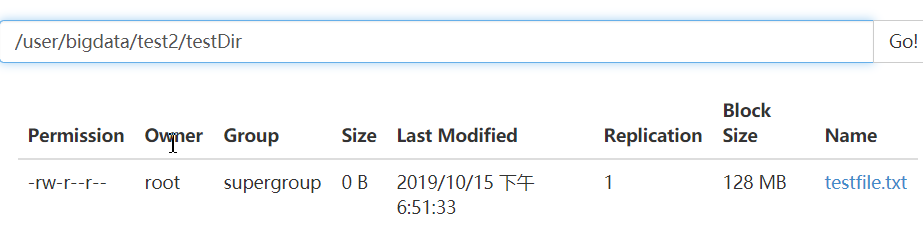
|  |
| --- |
| **package** com.fc.two;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.\*;  **import** java.io.\*;  **import** java.text.SimpleDateFormat;  **public** **class** TestTwo\_5 {  /\*\*  \* 给定HDFS中某一个目录， 输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，  \* 如果该文件是目录，则递归输出该目录下所有文件相关信息  \*/  **public** **static** **void** lookDir(Configuration conf, String dir) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  //获取该目录下，所有的文件 包括了递归输出  //生成一个迭代器  RemoteIterator<LocatedFileStatus> iterator = fs.listFiles(  **new** Path(dir), **true**);  //打印文件信息  **while** (iterator.hasNext()) {  LocatedFileStatus next = iterator.next();  System.***out***.println("读写权限: " + next.getPermission().toString());  System.***out***.println("文件大小: " + next.getLen());  //日期格式转换  **long** time = next.getModificationTime();  SimpleDateFormat format = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  System.***out***.println("时间: " + format.format(time));  System.***out***.println("存储路径: " + next.getPath().toString());  }  fs.close();  }  **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  //设置HDFS上要查看的目录路径  String dir = "/user/bigdata";  **try** {  *lookDir*(conf, dir);  } **catch** (IOException e) {  e.printStackTrace();  }  }  } |

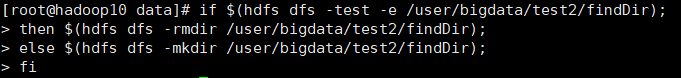


1. 提供一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录；
   1. shell命令
      1. 用到的HDFS命令
         1. 
      2. 创建
         1. 
      3. 删除
         1. 
   2. java代码

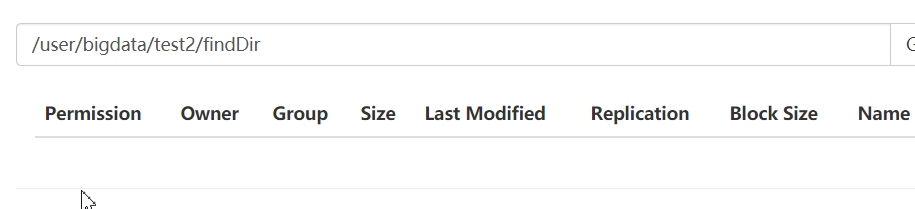
|  |
| --- |
| **package** com.fc.two;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.\*;  **import** java.io.\*;  /\*\*  \* 提供一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录  \* **@author** root  \*  \*/  **public** **class** TestTwo\_6 {    **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  conf.set("fs.default.name","hdfs://192.168.242.10:9000");  //指定要创建或者删除的目录或者文件  String file = "/user/bigdata/test2/testDir/testfile.txt";  String dir = "/user/bigdata/test2/testDir";  **try** {  //检查指定的路径是否存在  **if** (*findPath*(conf, dir)) {  System.***out***.println("指定路径存在，将尝试删除指定文件");  //如果存在，则删除该目录下的指定文件  *rm*(conf, file);  } **else** {  //如果不存在，则进行创建，并创建指定文件  System.***out***.println("路径不存在，将创建目录及文件");  *mkdir*(conf, dir);  *touchz*(conf, file);  }  } **catch** (Exception e) {  e.printStackTrace();  }  }    /\*\*  \* 判断路径是否存在  \*/  **public** **static** **boolean** findPath(Configuration conf, String dir) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  **return** fs.exists(**new** Path(dir));  }    /\*\*  \* 创建对应目录后，创建对应的文件  \*/  **public** **static** **void** touchz(Configuration conf, String file) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  FSDataOutputStream os = fs.create(**new** Path(file));  os.close();  fs.close();  }    /\*\*  \* 指定路径存在的情况下，尝试删除指定文件  \*/  **public** **static** **void** rm(Configuration conf, String file) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  fs.delete(**new** Path(file), **false**);  fs.close();  }    /\*\*  \* 指定路径不存在时，创建对应的目录  \*/  **public** **static** **void** mkdir(Configuration conf, String dir) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  fs.mkdirs(**new** Path(dir));  fs.close();  }  } |



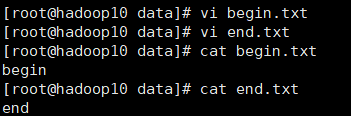


1. 提供一个HDFS的目录的路径，对该目录进行创建和删除操作。创建目录时，如果目录文件所在目录不存在，则自动创建相应目录；删除目录时，由用户指定当该目录不为空时是否还删除该目录；
   1. shell命令
      1. 
   2. java代码

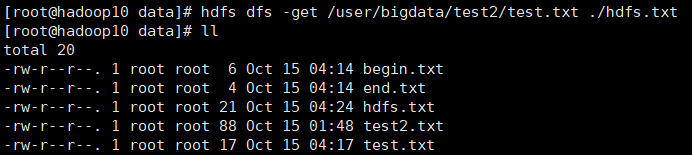
|  |
| --- |
| **package** com.fc.two;  **import** java.io.IOException;  **import** java.util.Scanner;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.LocatedFileStatus;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.fs.RemoteIterator;  /\*\*  \* 提供一个HDFS的目录的路径，对该目录进行创建和删除操作。  \* 如果该路径不存在，则自动创建该目录  \* 如果存在，则判断是否为空，并且将结果反馈给用户  \* 如果为空，自动删除  \* 如果不为空，由用户决定是否强制删除  \* **@author** root  \*  \*/  **public** **class** TestTwo\_7 {    **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  conf.set("fs.default.name","hdfs://192.168.242.10:9000");  String path = "/user/bigdata/test2/findDir"; // HDFS目录  **try** {  //判断目录是否存在  **if** (*testPath*(conf, path) ) {  //存在，继续判断是否为空，如果为空，自动删除  **if** (*isEmpty*(conf, path)) {  //自动删除  System.***out***.println("目录已经自动删除");  *rm*(conf, path);  }**else** {  //如果不为空，给出提示信息，由用户决定是否删除  System.***out***.println("该目录不为空，是否强制删除");  Scanner scanner=**new** Scanner(System.***in***);  String choose = scanner.next();  **if** (choose.equals("删除")) {  //强制删除  *rm\_force*(conf, path);  }  }  } **else** {  *mkdir*(conf, path);  }  } **catch** (Exception e) {  e.printStackTrace();  }  }    /\*\*  \* 判断路径是否存在  \*/  **public** **static** **boolean** testPath(Configuration conf, String path) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  **return** fs.exists(**new** Path(path));  }  /\*\*  \* 判断是否为空  \*/  **public** **static** **boolean** isEmpty(Configuration conf, String path) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  RemoteIterator<LocatedFileStatus> remoteIterator = fs.listFiles(**new** Path(path), **true**);  **return** !remoteIterator.hasNext();  }    /\*\*  \* 创建目录  \*/  **public** **static** **void** mkdir(Configuration conf, String path) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  Path dirPath = **new** Path(path);  fs.mkdirs(dirPath);  fs.close();  }    /\*\*  \* 删除目录  \*/  **public** **static** **void** rm(Configuration conf, String path) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  fs.delete(**new** Path(path), **true**);  fs.close();  }    /\*\*  \* 强制删除目录  \* **@throws** IOException  \*/  **public** **static** **void** rm\_force(Configuration conf,String path) **throws** IOException{  FileSystem fs = FileSystem.*get*(conf);  fs.delete(**new** Path(path), **true**);  fs.close();  }  } |

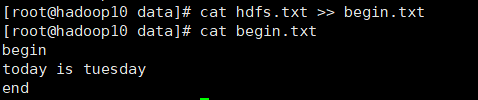




1. 向HDFS中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾；
   1. shell命令
      1. 准备两个文件,一个存放将要添加到HDFS中指定文件开头的内容,一个存放末尾内容
         1. 
      2. 向HDFS中/user/bigdata/test2/test.txt文件操作
         1. 查看原文件中内容
            1. 
         2. 追加到末尾
            1. 
            2. 
         3. 添加到开头
            1. 第一种方式

没有直接的命令可以实现添加在开头,所以可以将hdfs上的指定文件先下载下来,然后在追加开头,然后重新上传







追加成功



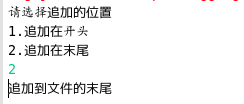
* + - * 1. 第二种方式

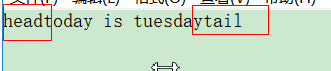
将HDFS上指定文件下载到本地,将要追加在开头的内容存储在一个文件A中并上传到HDFS上

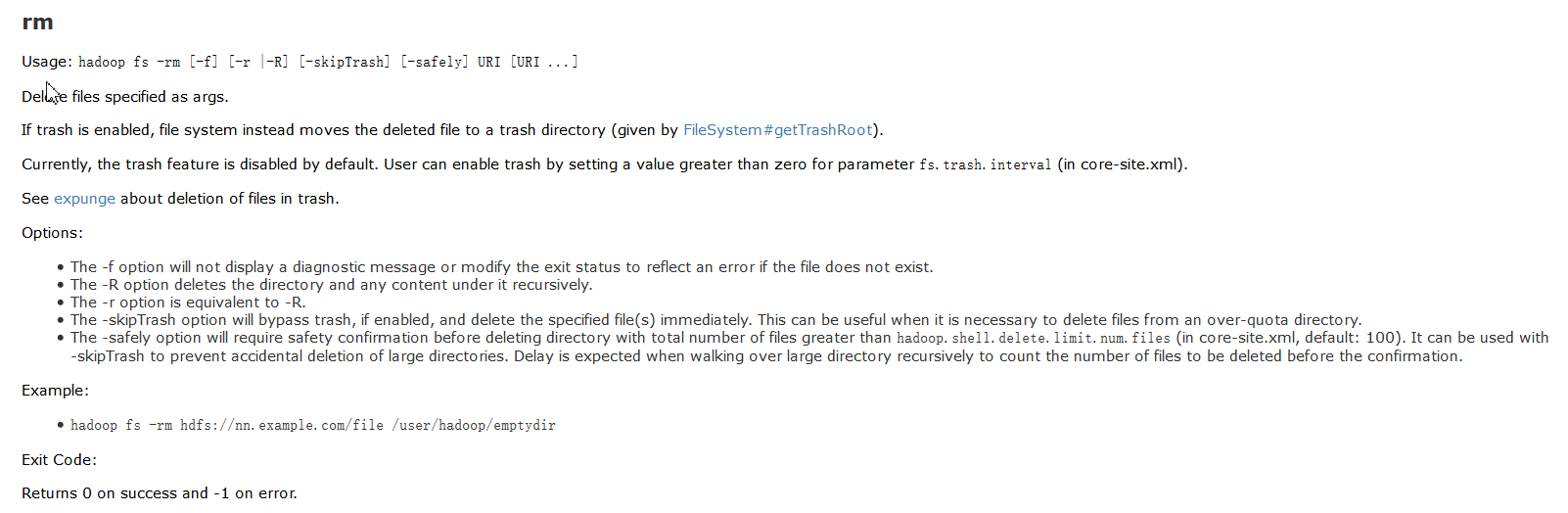
然后将最初的文件的内容追加在A文件之后

* 1. Java代码
     1. 这里使用javaAPI的话,可以不像Linux上命令操作那么麻烦,在代码中可以用变量保存要追加的内容,然后通过API选择头追加还是尾追加即可

|  |
| --- |
| package com.fc.two;  import java.io.FileInputStream;  import java.io.IOException;  import java.util.Scanner;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FSDataOutputStream;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  /\*\*  \* 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  \*  \* @author root  \*  \*/  /\*\*  \* 追加文本内容  \*/  public class TestTwo\_8 {    public static void main(String[] args) throws IOException {  //创建配置文件  Configuration conf = new Configuration();  conf.set("fs.default.name","hdfs://192.168.242.10:9000");  //如集群节点少于3个在运行时就会抛异常，解决方案，修改以下两个配置  conf.set("dfs.client.block.write.replace-datanode-on-failure.policy","NEVER");  conf.set("dfs.client.block.write.replace-datanode-on-failure.enable","true");  //HDFS中要添加内容的文件  String hdfsFile = "/user/bigdata/test2/test.txt";  String begin="head";  String end="tail";  Scanner scanner=new Scanner(System.in);  System.out.println("请选择追加的位置");  System.out.println("1.追加在开头");  System.out.println("2.追加在末尾");  int choose = scanner.nextInt();  switch (choose) {  case 1:  //1.将HDFS上的指定文件下载到Linux中  //指定本地下载的路径  String localPath = "/usr/local/bigdataTest/data/extra.txt";  //下载到本地  moveToLocalFile(conf, hdfsFile, localPath);  //2.在HDFS上新增同名文件，内容为追加在开头的内容  create(conf, hdfsFile);  //3.将下载到Linux本地的文件内容追加到刚刚创建的HDFS上的文件中  appendToTail(conf, begin, hdfsFile);  appendToFile(conf, localPath, hdfsFile);  System.out.println("追加到文件开头");  break;  case 2:  appendToTail(conf,end,hdfsFile);  System.out.println("追加到文件的末尾");  break;  }  }  /\*\*  \* 追加内容到文件末尾  \*/  public static void appendToTail(Configuration conf, String begin, String hdfsFile) throws IOException {  //创建文件输出流，将要追加的内容添加到HDFS上的文件中  FileSystem fs = FileSystem.get(conf);  FSDataOutputStream out = fs.append(new Path(hdfsFile));  out.write(begin.getBytes());  out.close();  fs.close();  }  /\*\*  \* 拷贝本地文件的内容，添加到HDFS对应文件后，追加操作  \*/  public static void appendToFile(Configuration conf, String localPath, String hdfsFile) throws IOException {  //通过流的方式进行拷贝  FileSystem fs = FileSystem.get(conf);  FileInputStream in = new FileInputStream(localPath);  FSDataOutputStream out = fs.append(new Path(hdfsFile));  byte[] data = new byte[1024];  int length = -1;  while ( (length = in.read(data)) > 0 ) {  out.write(data, 0, length);  }  out.close();  in.close();  fs.close();  }  /\*\*  \* 移动文件到本地  \*/  public static void moveToLocalFile(Configuration conf, String hdfsFile, String localPath) throws IOException {  FileSystem fs = FileSystem.get(conf);  fs.moveToLocalFile(new Path(hdfsFile), new Path(localPath));  }    /\*\*  \* 创建文件  \*/  public static void create(Configuration conf, String hdfsFile) throws IOException {  FileSystem fs = FileSystem.get(conf);  FSDataOutputStream os = fs.create(new Path(hdfsFile));  os.close();  fs.close();  }  } |





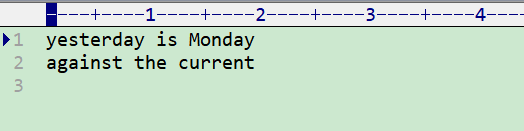
1. 删除HDFS中指定的文件；
   1. 用到的HDFS命令
      1. 
   2. shell命令
      1. 
   3. java代码

|  |
| --- |
| **package** com.fc.two;  **import** java.io.IOException;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.Path;  /\*\*  \* 删除HDFS中指定的文件  \* **@author** root  \*  \*/  **public** **class** TestTwo\_9 {    **public** **static** **void** main(String[] args) {  Configuration conf = **new** Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  String hdfsFile = "/user/bigdata/test2/test.txt"; // HDFS文件  **try** {  *rm*(conf, hdfsFile);  System.***out***.println("删除完毕");  } **catch** (Exception e) {  e.printStackTrace();  }  }    /\*\*  \* 删除指定文件  \*/  **public** **static** **void** rm(Configuration conf, String hdfsFile)  **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  fs.delete(**new** Path(hdfsFile), **false**);  fs.close();  }  } |



1. 在HDFS中，将文件从源路径移动到目的路径。
   1. 说明
      1. 这里设置的源路径为:/user/bigdata/test2/test.txt
      2. 目标路径为:/user/bigdata/test2/testDir/test.txt
   2. shell命令
      1. 
   3. java代码

|  |
| --- |
| **package** com.fc.two;  **import** java.io.IOException;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.Path;  **public** **class** TestTwo\_10 {    **public** **static** **void** main(String[] args) **throws** IOException {  Configuration conf = **new** Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  String source ="/user/bigdata/test2/test.txt";  String target="/user/bigdata/test2/testDir/test.txt";  *mv*(conf, source, target);  System.***out***.println("移动完成");  }    /\*\*  \* 移动文件  \*/  **public** **static** **void** mv(Configuration conf, String source,  String target) **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  fs.rename(**new** Path(source), **new** Path(target));  fs.close();  }  } |

1. 编程实现一个类“MyFSDataInputStream”，该类继承“org.apache.hadoop.fs.FSDataInputStream”，要求如下：实现按行读取HDFS中指定文件的方法“readLine()”，如果读到文件末尾，则返回空，否则返回文件一行的文本。
2. 待读取的文件
   * 1. 
3. java代码

|  |
| --- |
| **package** com.fc.two;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.FSDataInputStream;  **import** org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.Path;  **public** **class** MyFSDataInputStream **extends** FSDataInputStream {  **public** MyFSDataInputStream(InputStream in) {  **super**(in);  }  **public** **static** **void** main(String[] args) **throws** IOException {  Configuration conf = **new** Configuration();  conf.set("fs.default.name", "hdfs://192.168.242.10:9000");  String hdfsFile = "/user/bigdata/test2/test.txt";  *read*(conf, hdfsFile);  }    **public** **static** **void** read(Configuration conf, String hdfsFile)  **throws** IOException {  FileSystem fs = FileSystem.*get*(conf);  //打开输入流  FSDataInputStream in = fs.open(**new** Path(hdfsFile));  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(in));  String line = **null**;  //按行读取  **while** ((line = br.readLine()) != **null**) {  System.***out***.println(line);  }  fs.close();  }  } |

1. 控制台输出
   1. 
2. 查看Java帮助手册或其它资料，用“java.net.URL”和“org.apache.hadoop.fs.FsURLStreamHandlerFactory”编程完成输出HDFS中指定文件的文本到终端中。

java代码:

|  |
| --- |
| **package** com.fc.two;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.net.MalformedURLException;  **import** java.net.URL;  **import** org.apache.hadoop.fs.FsUrlStreamHandlerFactory;  **import** org.apache.hadoop.io.IOUtils;  **public** **class** TestFsURL {  **public** **static** **void** main(String[] args) {  URL.*setURLStreamHandlerFactory*(**new** FsUrlStreamHandlerFactory());  InputStream in=**null**;  **try** {  in=**new** URL("hdfs://192.168.242.10:9000/user/bigdata/test2/test.txt").openStream();  IOUtils.*copyBytes*(in, System.***out***, 2048,**false**);  } **catch** (MalformedURLException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }**finally**{  IOUtils.*closeStream*(in);  }  }  } |

输出结果



实验总结:

1. 整个实验做完之后,分析代码的流程,无非就是三步,获取hdfs的客户端对象--->执行对应的操作(在这一步中,可能涉及到对文件目录等的不同操作)-->关闭资源
2. 整体难度不是很大,做每一道题的时候,关键在于找到执行的核心操作,以及查找到对应的hdfs调用的API,然后按部就班根据流程写出来代码就可以了,shell命令见名知义,设计的很好,虽然一些命令的参数比较多,但是结合文档的注释还是不难解决的.