（答案）

实验 2

第三章 熟悉常用的 HDFS 操作

宋 曼

目录

1 实验目的.....1
2 实验平台.....1
3 实验内容和要求.....1

实验 2

第三章 熟悉常用的 HDFS 操作

1 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。

2 实验平台

操作系统：Linux
Hadoop 版本：2.6.0 或以上版本
JDK 版本：1.6 或以上版本
Java IDE：Eclipse

3 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：

提示：
1) 部分 Shell 命令的参数路径只能是本地路径或者 HDFS 路径。 2) 若 Shell 命令的参数既可以是本地路径，也可以是 HDFS 路径时，务必注意区分。为保证操作正确，可指定路径前缀 hdfs:/// 或者 file:/// 3) 注意区分相对路径与绝对路径

（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；

Shell 命令: 检查文件是否存在: hdfs dfs -test -e text.txt(执行完这一句不会输出结果，需要继续输入命令 " echo $?") 追加命令: hdfs dfs -appendToFile local.txt text.txt 覆盖命令 1: hdfs dfs -copyFromLocal -f local.txt text.txt 覆盖命令 2: hdfs dfs -cp -f file:///home/hadoop/local.txt text.txt 也可以使用如下命令实现： （如下代码可视为一行代码，在终端中输入第一行代码后，直到输入 fi 才会真正执行）： if $(hdfs dfs -test -e text.txt); then $(hdfs dfs -appendToFile local.txt text.txt); else $(hdfs dfs -copyFromLocal -f local.txt text.txt); fi

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
    /**
     * 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 */
        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();
      }

   /**
    * 主函数
    */
   public static void main(String[] args) {
      Configuration conf = new Configuration();
      conf.set("fs.default.name","hdfs://localhost:9000");
      String localFilePath = "/home/hadoop/text.txt";    // 本地路径
      String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 路径
      String choice = "append";    // 若文件存在则追加到文件末尾
//    String choice = "overwrite";    // 若文件存在则覆盖

      try {
         /* 判断文件是否存在  */
         Boolean fileExists = false;
         if (HDFSApi.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) { // 文件不存在，则上传
            HDFSApi.copyFromLocalFile(conf, localFilePath, remoteFilePath);
            System.out.println(localFilePath + " 已上传至  " + remoteFilePath);
         } else if ( choice.equals("overwrite") ) {    // 选择覆盖
            HDFSApi.copyFromLocalFile(conf, localFilePath, remoteFilePath);
            System.out.println(localFilePath + " 已覆盖  " + remoteFilePath);
         } else if ( choice.equals("append") ) {   // 选择追加
            HDFSApi.appendToFile(conf, localFilePath, remoteFilePath);
            System.out.println(localFilePath + " 已追加至  " + remoteFilePath);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
```

```
    }
}
```

（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；

Shell 命令：

```
if $(hdfs dfs -test -e file:///home/hadoop/text.txt);
then $(hdfs dfs -copyToLocal text.txt ./text2.txt);
else $(hdfs dfs -copyToLocal text.txt ./text.txt);
fi
```

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
    /**
     * 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.");
                Integer i = 0;
                while (true) {
                    f = new File(localFilePath + "_" + i.toString());
                    if (!f.exists()) {
                        localFilePath = localFilePath + "_" + i.toString();
                        break;
                    }
                }
                System.out.println("将重新命名为: " + localFilePath);
            }

            // 下载文件到本地
            Path localPath = new Path(localFilePath);
            fs.copyToLocalFile(remotePath, localPath);
```

```
            fs.close();
        }

    /**
     * 主函数
     */
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        conf.set("fs.default.name","hdfs://localhost:9000");
        String localFilePath = "/home/hadoop/text.txt";    // 本地路径
        String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 路径

        try {
            HDFSApi.copyToLocal(conf, remoteFilePath, localFilePath);
            System.out.println("下载完成");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

（3） 将 HDFS 中指定文件的内容输出到终端中；

Shell 命令：

```
hdfs dfs -cat text.txt
```

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
    /**
     * 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();
        }

    /**
     * 主函数
     */
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        conf.set("fs.default.name","hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 路径

        try {
            System.out.println("读取文件: " + remoteFilePath);
            HDFSApi.cat(conf, remoteFilePath);
            System.out.println("\n 读取完成");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

（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；

Shell 命令：

```
hdfs dfs -ls -h text.txt
```

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;
import java.text.SimpleDateFormat;

public class HDFSApi {
    /**
     * 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);
        FileStatus[] fileStatuses = fs.listStatus(remotePath);
```

```
            for (FileStatus s : fileStatuses) {
                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.default.name","hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 路径

        try {
            System.out.println("读取文件信息: " + remoteFilePath);
            HDFSApi.ls(conf, remoteFilePath);
            System.out.println("\n 读取完成");
        } catch (Exception 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；

Shell 命令：

```
hdfs dfs -ls -R -h /user/hadoop
```

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;
import java.text.SimpleDateFormat;
```

```
public class HDFSApi {
    /**
     * 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);
            System.out.println();
        }
        fs.close();
    }

  /**
   * 主函数
   */
  public static void main(String[] args) {
    Configuration conf = new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");
    String remoteDir = "/user/hadoop";    // HDFS 路径

    try {
      System.out.println("(递归)读取目录下所有文件的信息: " + remoteDir);
      HDFSApi.lsDir(conf, remoteDir);
      System.out.println("读取完成");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

（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；

Shell 命令：

```
if $(hdfs dfs -test -d dir1/dir2);
then $(hdfs dfs -touchz dir1/dir2/filename);
else $(hdfs dfs -mkdir -p dir1/dir2 && hdfs dfs -touchz dir1/dir2/filename);
fi
```

删除文件：hdfs dfs -rm dir1/dir2/filename

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
    /**
     * 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) {
    Configuration conf = new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");
    String remoteFilePath = "/user/hadoop/input/text.txt";    // HDFS 路径
    String remoteDir = "/user/hadoop/input";    // HDFS 路径对应的目录

    try {
      /* 判断路径是否存在，存在则删除，否则进行创建 */
      if ( HDFSApi.test(conf, remoteFilePath) ) {
        HDFSApi.rm(conf, remoteFilePath); // 删除
        System.out.println("删除路径: " + remoteFilePath);
      } else {
        if ( !HDFSApi.test(conf, remoteDir) ) { // 若目录不存在，则进行创建
          HDFSApi.mkdir(conf, remoteDir);
          System.out.println("创建文件夹: " + remoteDir);
        }
        HDFSApi.touchz(conf, remoteFilePath);
        System.out.println("创建路径: " + remoteFilePath);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

（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；

Shell 命令：

创建目录：hdfs dfs -mkdir -p dir1/dir2
删除目录（如果目录非空则会提示 not empty，不执行删除）：hdfs dfs -rmdir dir1/dir2
强制删除目录：hdfs dfs -rm -R dir1/dir2

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
    /**
     * 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递归删除所有文件 */
        boolean result = fs.delete(dirPath, true);
        fs.close();
        return result;
    }

  /**
   * 主函数
   */
  public static void main(String[] args) {
    Configuration conf = new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");
    String remoteDir = "/user/hadoop/input";    // HDFS 目录
    Boolean forceDelete = false;  // 是否强制删除

    try {
      /* 判断目录是否存在，不存在则创建，存在则删除 */
      if ( !HDFSApi.test(conf, remoteDir) ) {
        HDFSApi.mkdir(conf, remoteDir); // 创建目录
        System.out.println("创建目录: " + remoteDir);
      } else {
        if ( HDFSApi.isDirEmpty(conf, remoteDir) || forceDelete ) { // 目录为空或强制删除
          HDFSApi.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);
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

（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；

Shell 命令：

追加到文件末尾：hdfs dfs -appendToFile local.txt text.txt
追加到文件开头：
（由于没有直接的命令可以操作，方法之一是先移动到本地进行操作，再进行上传覆盖）：
hdfs dfs -get text.txt
cat text.txt >> local.txt
hdfs dfs -copyFromLocal -f text.txt text.txt

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
    /**
     * 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) {
        Configuration conf = new Configuration();
        conf.set("fs.default.name","hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 文件
        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

        try {
            /* 判断文件是否存在 */
            if ( !HDFSApi.test(conf, remoteFilePath) ) {
                System.out.println("文件不存在: " + remoteFilePath);
            } else {
                if ( choice.equals("after") ) { // 追加在文件末尾
                    HDFSApi.appendContentToFile(conf, content, remoteFilePath);
                    System.out.println("已追加内容到文件末尾" + remoteFilePath);
                } else if ( choice.equals("before") ) { // 追加到文件开头
                    /* 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 = "/user/hadoop/tmp.txt";
                    HDFSApi.moveToLocalFile(conf, remoteFilePath, localTmpPath); // 移动到本地
                    HDFSApi.touchz(conf, remoteFilePath);    // 创建一个新文件
                    HDFSApi.appendContentToFile(conf, content, remoteFilePath);  // 先写入新内容
                    HDFSApi.appendToFile(conf, localTmpPath, remoteFilePath);  // 再写入原来内容
                    System.out.println("已追加内容到文件开头: " + remoteFilePath);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

（9） 删除 HDFS 中指定的文件；

Shell 命令：

```
hdfs dfs -rm text.txt
```

Java 命令：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
```

```
    /**
     * 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) {
    Configuration conf = new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");
    String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 文件

    try {
      if ( HDFSApi.rm(conf, remoteFilePath) ) {
        System.out.println("文件删除: " + remoteFilePath);
      } else {
        System.out.println("操作失败（文件不存在或删除失败）");
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

（10） 删除 HDFS 中指定的目录，由用户指定目录中如果存在文件时是否删除目录；

Shell 命令：

删除目录（如果目录非空则会提示 not empty，不执行删除）：hdfs dfs -rmdir dir1/dir2
强制删除目录：hdfs dfs -rm -R dir1/dir2

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
```

```
    /**
     * 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();
    }

    /**
     * 删除目录
     */
    public static boolean rmDir(Configuration conf, String remoteDir, boolean recursive) throws
IOException {
        FileSystem fs = FileSystem.get(conf);
        Path dirPath = new Path(remoteDir);
        /* 第二个参数表示是否递归删除所有文件 */
        boolean result = fs.delete(dirPath, recursive);
        fs.close();
        return result;
    }

  /**
   * 主函数
   */
  public static void main(String[] args) {
    Configuration conf = new Configuration();
    conf.set("fs.default.name","hdfs://localhost:9000");
    String remoteDir = "/user/hadoop/input";    // HDFS 目录
    Boolean forceDelete = false;  // 是否强制删除

    try {
      if ( !HDFSApi.isDirEmpty(conf, remoteDir) && !forceDelete ) {
        System.out.println("目录不为空，不删除");
      } else {
        if ( HDFSApi.rmDir(conf, remoteDir, forceDelete) ) {
          System.out.println("目录已删除: " + remoteDir);
        } else {
          System.out.println("操作失败");
        }
```

```
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
}
```

（11） 在 HDFS 中，将文件从源路径移动到目的路径。

Shell 命令：

```
hdfs dfs -mv text.txt text2.txt
```

Java 代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSApi {
      /**
       * 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);
            boolean result = fs.rename(srcPath, dstPath);
            fs.close();
            return result;
      }

   /**
    * 主函数
    */
   public static void main(String[] args) {
      Configuration conf = new Configuration();
      conf.set("fs.default.name","hdfs://localhost:9000");
      String remoteFilePath = "hdfs:///user/hadoop/text.txt";    // 源文件 HDFS 路径
      String remoteToFilePath = "hdfs:///user/hadoop/new.txt";    // 目的 HDFS 路径

      try {
        if ( HDFSApi.mv(conf, remoteFilePath, remoteToFilePath) ) {
          System.out.println("将文件  " + remoteFilePath + " 移动到  " + remoteToFilePath);
        } else {
```

```
                System.out.println("操作失败(源文件不存在或移动失败)");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2. 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。

Java 代码：

```
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.*;

public class MyFSDataInputStream extends FSDataInputStream {
    public MyFSDataInputStream(InputStream in) {
        super(in);
    }

    /**
     * 实现按行读取
     * 每次读入一个字符，遇到"\n"结束，返回一行内容
     */
    public static String readline(BufferedReader br) throws IOException {
        char[] data = new char[1024];
        int read = -1;
        int off = 0; // 循环执行时，br 每次会从上一次读取结束的位置继续读取，因此该函数里，off 每次都从 0 开始
        while ( (read = br.read(data, off, 1)) != -1 ) {
            if (String.valueOf(data[off]).equals("\n") ) {
                off += 1;
                break;
            }
            off += 1;
        }

        if (off > 0) {
            return String.valueOf(data);
        } else {
```

```
            return null;
        }
    }

    /**
     * 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 br = new BufferedReader(new InputStreamReader(in));
        String line = null;
        while ( (line = MyFSDataInputStream.readline(br)) != null ) {
            System.out.println(line);
        }
        br.close();
        in.close();
        fs.close();
    }

    /**
     * 主函数
     */
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        conf.set("fs.default.name","hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/text.txt";    // HDFS 路径
        try {
            MyFSDataInputStream.cat(conf, remoteFilePath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3. 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：

```
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.IOUtils;
import java.io.*;
```

```
import java.net.URL;

public class HDFSApi {
   static{
            URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());
      }

   /**
    * 主函数
    */
   public static void main(String[] args) throws Exception {
      String remoteFilePath = "hdfs:///user/hadoop/text.txt";    // HDFS 文件
      InputStream in = null;
            try{
                  /* 通过 URL 对象打开数据流，从中读取数据 */
                  in = new URL(remoteFilePath).openStream();
                  IOUtils.copyBytes(in,System.out,4096,false);
            } finally{
                  IOUtils.closeStream(in);
            }
   }
}
```