# 实　　验　　报　　告

| | | | | | |
|---|---|---|---|---|---|
| 学　　号 | 19020011038 | 姓　　名 | 岳宇轩 | 专业班级 | 2019 级慧与卓越工程师班 |
| 课程名称 | 大数据导论 | | | 学期 | 2022 年秋季学期 |
| 任课教师 | 刘洁 刘艳艳 | 完成日期 | 2022.10.24 | 上机课时间 | 周一 56 节（双周） |
| 实 验 名 称 | 实验三、熟悉常用的 HDFS 操作 | | | | |

## 一、实验要求（10%）

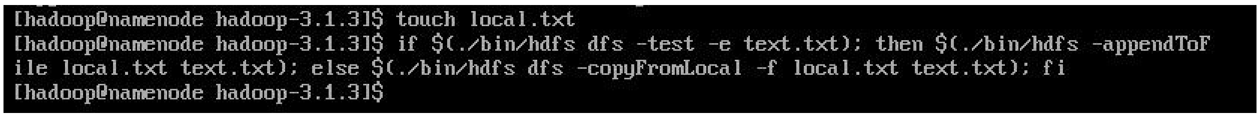
1. 使用 Hadoop 提供的 Shell 命令完成下列任务,使用 JAVA API 编程实现第（1）题：
提示：
a) 部分 Shell 命令的参数路径只能是本地路径或者 HDFS 路径。
b) 若 Shell 命令的参数既可以是本地路径，也可以是 HDFS 路径时，务必注意区分。
为保证操作正确，可指定路径前缀 hdfs:/// 或者 file:///
c) 注意区分相对路径与绝对路径。
d)具体命令的说明可参考教材或 http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html

(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；(用 JAVA 编程实现相同功能)
(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；
(3) 将 HDFS 中指定文件的内容输出到终端中；
(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；
(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；
(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；
(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；
(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；
(9) 删除 HDFS 中指定的文件；
(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；
(11) 在 HDFS 中，将文件从源路径移动到目的路径。
(12)

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。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。
3.
4. 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。

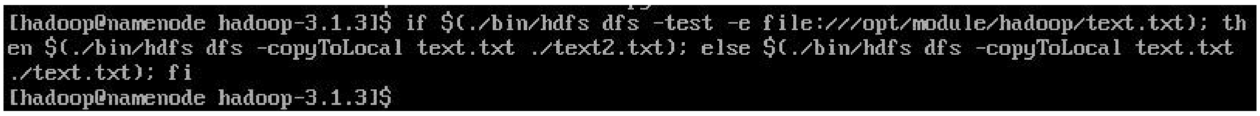
## 二、实验内容及步骤（80%）

1.（1）



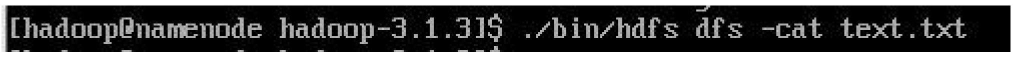
```
[hadoop@namenode hadoop-3.1.3]$ touch local.txt
[hadoop@namenode hadoop-3.1.3]$ if $(./bin/hdfs dfs -test -e text.txt); then $(./bin/hdfs -appendToF
ile local.txt text.txt); else $(./bin/hdfs dfs -copyFromLocal -f local.txt text.txt); fi
[hadoop@namenode hadoop-3.1.3]$
```

1.（2）



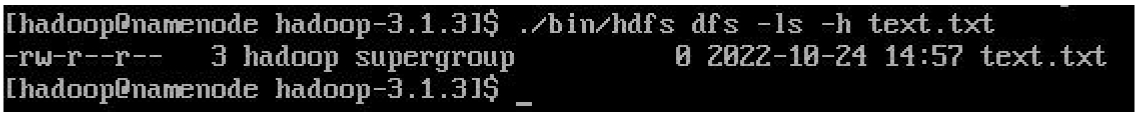
```
[hadoop@namenode hadoop-3.1.3]$ if $(./bin/hdfs dfs -test -e file:///opt/module/hadoop/text.txt); th
en $(./bin/hdfs dfs -copyToLocal text.txt ./text2.txt); else $(./bin/hdfs dfs -copyToLocal text.txt
./text.txt); fi
[hadoop@namenode hadoop-3.1.3]$
```

1.（3）



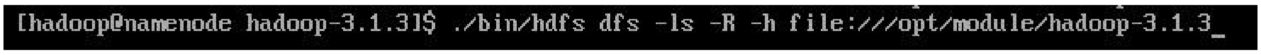
```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -cat text.txt
```

1.（4）

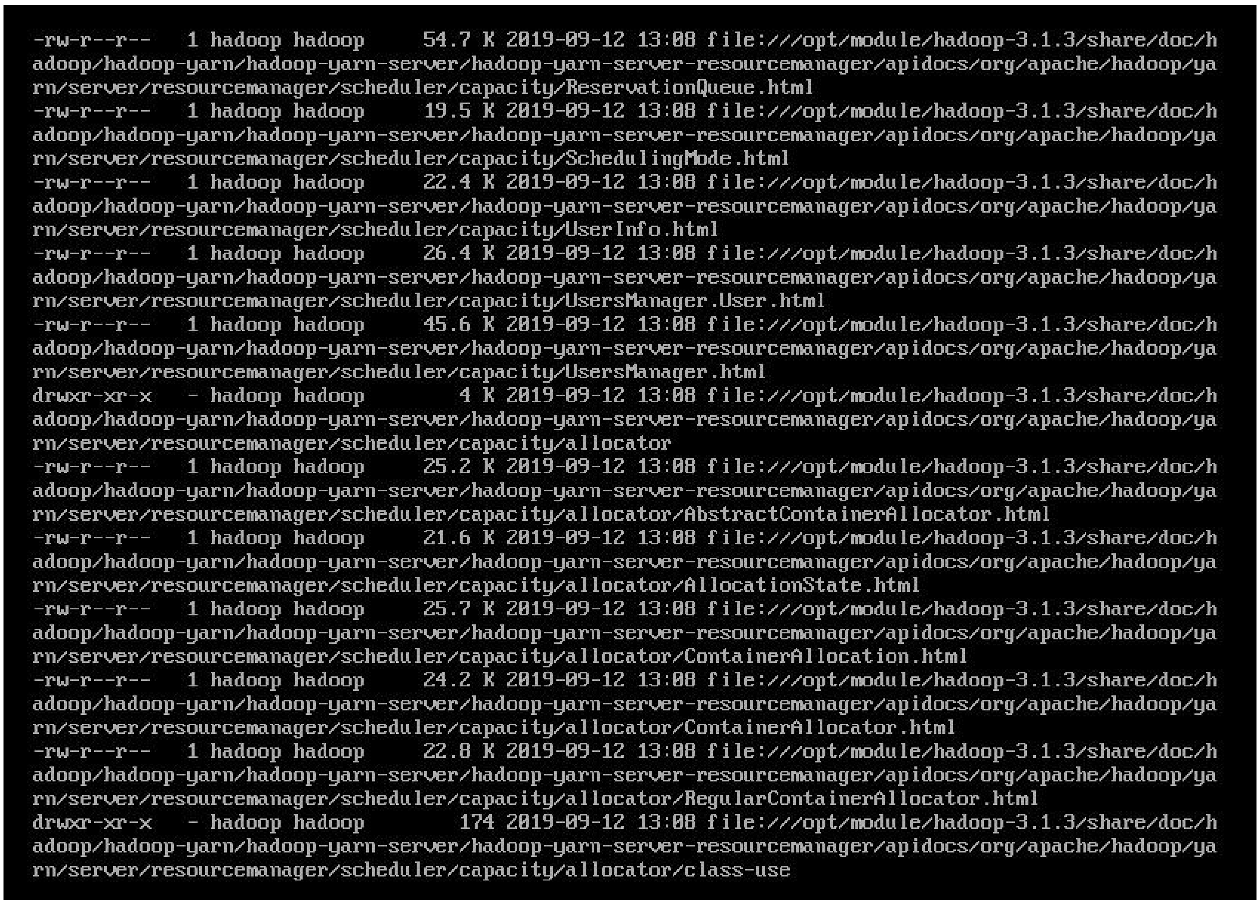


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -ls -h text.txt
-rw-r--r--   3 hadoop supergroup          0 2022-10-24 14:57 text.txt
[hadoop@namenode hadoop-3.1.3]$ _
```

1.（5）

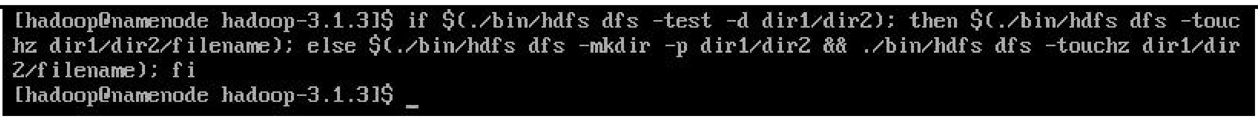


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -ls -R -h file:///opt/module/hadoop-3.1.3_
```

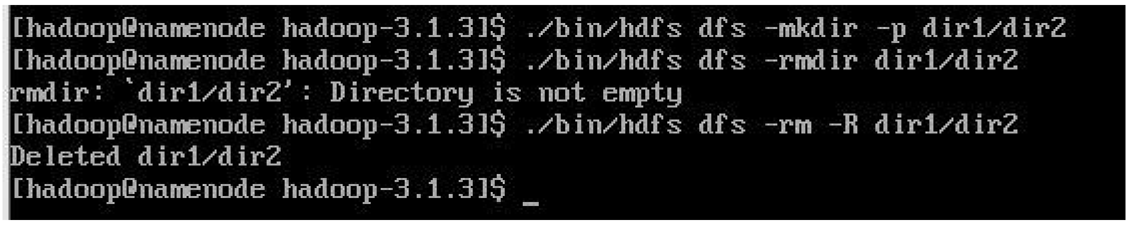


```
-rw-r--r--   1 hadoop hadoop     54.7 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/ReservationQueue.html
-rw-r--r--   1 hadoop hadoop     19.5 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/SchedulingMode.html
-rw-r--r--   1 hadoop hadoop     22.4 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/UserInfo.html
-rw-r--r--   1 hadoop hadoop     26.4 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/UsersManager.User.html
-rw-r--r--   1 hadoop hadoop     45.6 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/UsersManager.html
drwxr-xr-x   - hadoop hadoop        4 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator
-rw-r--r--   1 hadoop hadoop     25.2 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator/AbstractContainerAllocator.html
-rw-r--r--   1 hadoop hadoop     21.6 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator/AllocationState.html
-rw-r--r--   1 hadoop hadoop     25.7 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator/ContainerAllocation.html
-rw-r--r--   1 hadoop hadoop     24.2 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator/ContainerAllocator.html
-rw-r--r--   1 hadoop hadoop     22.8 K 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator/RegularContainerAllocator.html
drwxr-xr-x   - hadoop hadoop        174 2019-09-12 13:08 file:///opt/module/hadoop-3.1.3/share/doc/h
adoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apache/hadoop/ya
rn/server/resourcemanager/scheduler/capacity/allocator/class-use
```

1.（6）

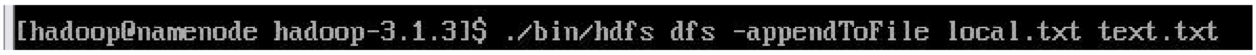

```
[hadoop@namenode hadoop-3.1.3]$ if $(./bin/hdfs dfs -test -d dir1/dir2); then $(./bin/hdfs dfs -touc
hz dir1/dir2/filename); else $(./bin/hdfs dfs -mkdir -p dir1/dir2 && ./bin/hdfs dfs -touchz dir1/dir
2/filename); fi
[hadoop@namenode hadoop-3.1.3]$ _
```

1. （7）

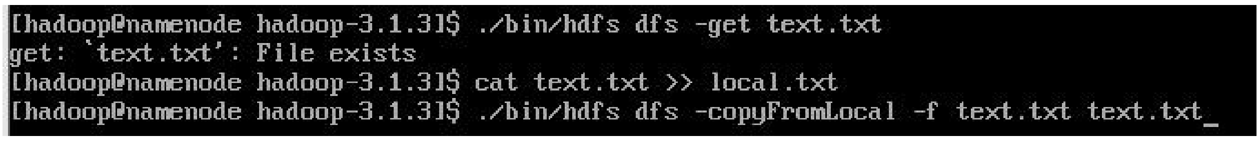

```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -mkdir -p dir1/dir2
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -rmdir dir1/dir2
rmdir: `dir1/dir2': Directory is not empty
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -rm -R dir1/dir2
Deleted dir1/dir2
[hadoop@namenode hadoop-3.1.3]$ _
```

1. （8）

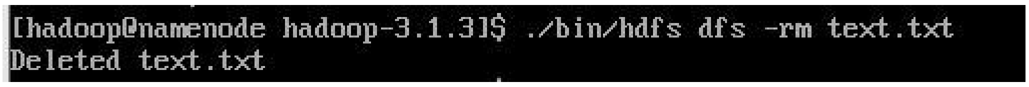
追加到文件末尾：


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -appendToFile local.txt text.txt
```

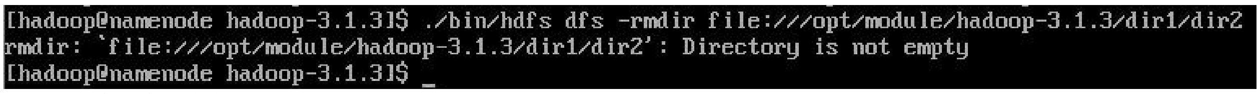
追加到文件开头


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -get text.txt
get: `text.txt': File exists
[hadoop@namenode hadoop-3.1.3]$ cat text.txt >> local.txt
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -copyFromLocal -f text.txt text.txt_
```

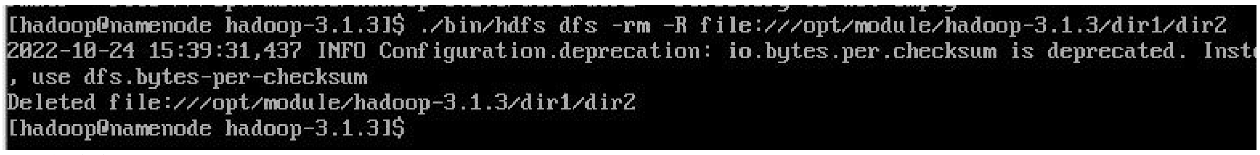
1. （9）


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -rm text.txt
Deleted text.txt
```

1. （10）

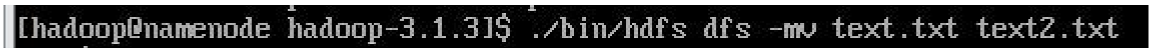

```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -rmdir file:///opt/module/hadoop-3.1.3/dir1/dir2
rmdir: `file:///opt/module/hadoop-3.1.3/dir1/dir2': Directory is not empty
[hadoop@namenode hadoop-3.1.3]$ _
```

非空无法直接删除


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -rm -R file:///opt/module/hadoop-3.1.3/dir1/dir2
2022-10-24 15:39:31,437 INFO Configuration.deprecation: io.bytes.per.checksum is deprecated. Inst
, use dfs.bytes-per-checksum
Deleted file:///opt/module/hadoop-3.1.3/dir1/dir2
[hadoop@namenode hadoop-3.1.3]$
```

强制删除

1. （11）


```
[hadoop@namenode hadoop-3.1.3]$ ./bin/hdfs dfs -mv text.txt text2.txt
```

2.

按照实验教程中的代码运行，可以得出如下的结果：

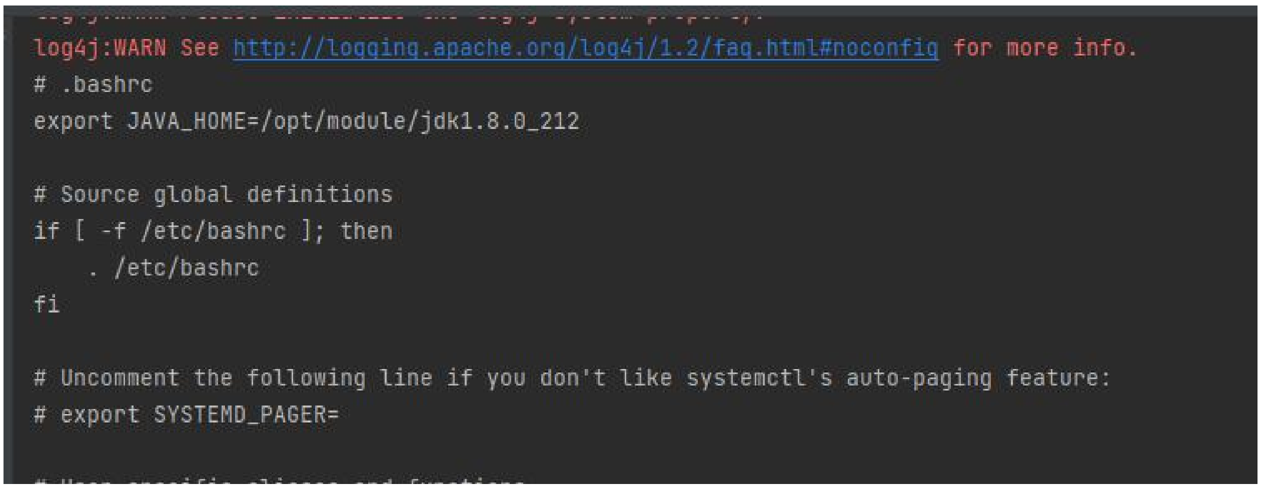
```
# .bashrc

export JAVA_HOME=/opt/module/jdk1.8.0_212



# Source global definitions

if [ -f /etc/bashrc ]; then

        . /etc/bashrc
```

```
fi



# Uncomment the following line if you don't like systemctl's auto-paging feature:

# export SYSTEMD_PAGER=



# User specific aliases and functions

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:    21.733 s
[INFO] Finished at: 2022-10-24T17:26:00+08:00
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

4.
新建 HDFSapi 类
使用如下代码：

```java
package com.ben.hdfs;

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://namenode:8020/user/hadoop/test/.bashrc"; // 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);
        }
    }
}
```

得到实验结果如图：



```
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
# .bashrc
export JAVA_HOME=/opt/module/jdk1.8.0_212

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=
```

5. 使用 Hadoop 提供的 Shell 命令完成下列任务，使用 JAVA API 编程实现第（1）题：

使用如下代码：

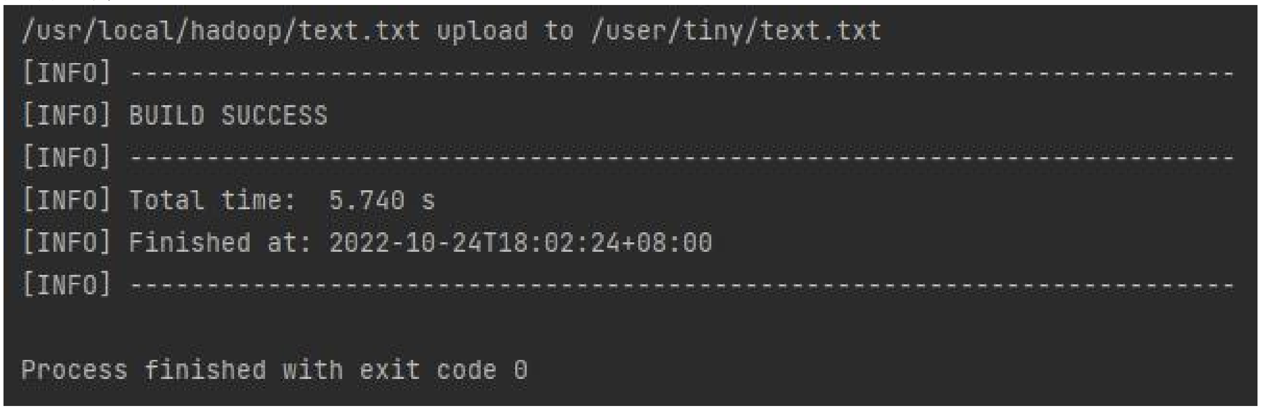
```java
package com.ben.hdfs;



import java.io.FileInputStream;

import java.io.IOException;



import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;
```

```java
public class CopyFromLocalFile {

    /**

     * 判断路径是否存在

     */

    public static boolean test(Configuration conf, String path) {

        try (FileSystem fs = FileSystem.get(conf)) {

            return fs.exists(new Path(path));

        } catch (IOException e) {

            e.printStackTrace();

            return false;

        }

    }



    /**

     * 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) {

        Path localPath = new Path(localFilePath);

        Path remotePath = new Path(remoteFilePath);
    try (FileSystem fs = FileSystem.get(conf)) {

        /* 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);

    } catch (IOException e) {

        e.printStackTrace();

    }


}


/**

 * 追加文件内容

 */

public static void appendToFile(Configuration conf, String localFilePath,

                                String remoteFilePath) {

    Path remotePath = new Path(remoteFilePath);

    try (FileSystem fs = FileSystem.get(conf);

         FileInputStream in = new FileInputStream(localFilePath);) {

        FSDataOutputStream out = fs.append(remotePath);

        byte[] data = new byte[1024];

        int read = -1;

        while ((read = in.read(data)) > 0) {
                out.write(data, 0, read);
            }
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

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

```java
boolean fileExists = false;

if (CopyFromLocalFile.test(conf, remoteFilePath)) {

    fileExists = true;

    System.out.println(remoteFilePath + " exist.");

} else {

    System.out.println(remoteFilePath + " not exist.");

}

/* 进行处理 */

if (!fileExists) { // 文件不存在，则上传

    CopyFromLocalFile.copyFromLocalFile(conf, localFilePath,

            remoteFilePath);

    System.out.println(localFilePath + " upload to " + remoteFilePath);

} else if (choice.equals("overwrite")) { // 选择覆盖

    CopyFromLocalFile.copyFromLocalFile(conf, localFilePath,

            remoteFilePath);

    System.out.println(localFilePath + " overwrite " + remoteFilePath);

} else if (choice.equals("append")) { // 选择追加

    CopyFromLocalFile.appendToFile(conf, localFilePath,

            remoteFilePath);

    System.out.println(localFilePath + " append to " + remoteFilePath);

}
```

```
        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

成功上传：



```
/usr/local/hadoop/text.txt upload to /user/tiny/text.txt
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.740 s
[INFO] Finished at: 2022-10-24T18:02:24+08:00
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

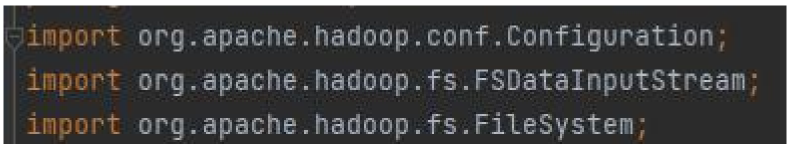
**三、心得总结（写出自己在完成实验过程中遇到的问题、解决方法，以及体会、收获等）（10%）**

通过本次实验，我熟悉了 HDFS 提供的 Shell 命令，包括对文件和文件夹的各种操作。

我认为本次实验的难点在于 JAVA 编程实现 Shell 命令，首先我遇到的第一个问题就是无法正确编译 pom.xml 文件，因为我自己电脑上之间就装过 maven，所以一开始就直接用了，后来发现不大行，又去官网上下了最新的 3.8.6 版本的 maven，还是不行。

后来通过去网上翻阅资料得知可能是 maven 依赖没有正确安装的原因。在 Idea 里找到更新 maven 依赖的地方进行更新之后，pom.xml 就没有标红了。

再就是 HDFSapi 文件也会有报红，就是最开始 import 这几个：



```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
```

都是标红的，也是费了比较多功夫才成功把他们安装好了。

由此，第一个编程任务就可以顺利完成了。

第二个编程任务中，值得注意的一点是 **HDFS 路径的修改**，使用第一问中的路径：

```java
String remoteFilePath =        "/user/hadoop/test/.bashrc";     // HDFS 路径
```

会报错 Process exited with an error: 1(Exit value: 1)。

网上很多博客说都是端口进程占用的问题，但是我对此做了排查之后，发现并非由此造成。仔细阅读报错信息，我认为是读取文件路径的问题，因此参考了前一问代码中的端口号和路径修改后完成了第二问：

```java
String remoteFilePath = "hdfs://namenode:8020/user/hadoop/test/.bashrc"; // HDFS 文件
```

总之，我认为这次实验的难度相较于以往都更加有挑战性，需要细致，耐心，要对 hdfs 有较好的了解，要学会配环境，对 JAVA 编程能力也是有十分的考验。此次实验可以说是令我获益匪浅。