数据科学导论实验报告

实验二 熟悉常用的 HDFS 操作

学　　生：

年　　级：2021 级

专　　业：软件工程

重庆大学大数据与软件学院

2023 年　10 月　　13 日

一、 实验目的：

1. 理解 HDFS 在 Hadoop 体系结构中的角色：通过实验，学习者将深入理解 HDFS 在 Hadoop 生态系统中的重要性和角色。HDFS 是 Hadoop 的分布式文件系统，用于存储和管理大数据，因此掌握其操作对于整个 Hadoop 平台的理解至关重要；

2. 熟练使用 HDFS 操作常用的 Shell 命令：实验的目的是帮助学习者熟练掌握 Hadoop Shell 命令，这些命令是与 HDFS 交互的关键工具。学习者将学会如何上传、下载、查看、创建、删除、移动等 HDFS 文件和目录，这些命令对于大数据处理和数据管理非常重要；

3. 熟悉 HDFS 操作常用的 Java API：实验不仅涉及 Shell 命令，还要求学习者使用 Java 编程语言来执行 HDFS 操作。通过编程实践，学习者将熟悉 Hadoop 提供的 Java API，了解如何在自己的应用程序中集成 HDFS 操作，以实现更高级的数据处理任务。这有助于学习者进一步深入学习大数据应用开发和数据分析。

二、 实验要求

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：

(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；

(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。

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。

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。

三、 开发环境：

- Lenovo Legion R9000P2021H
- VMware-workstation-full-17.0.0
- Bbuntu Kylin 16.04
- Eclipse-4.7.0-linux.gtk.x86_64

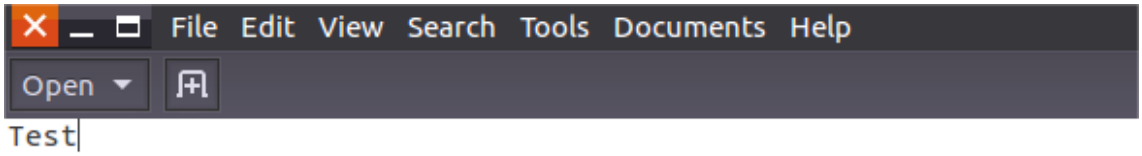
四、 实验内容：

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：

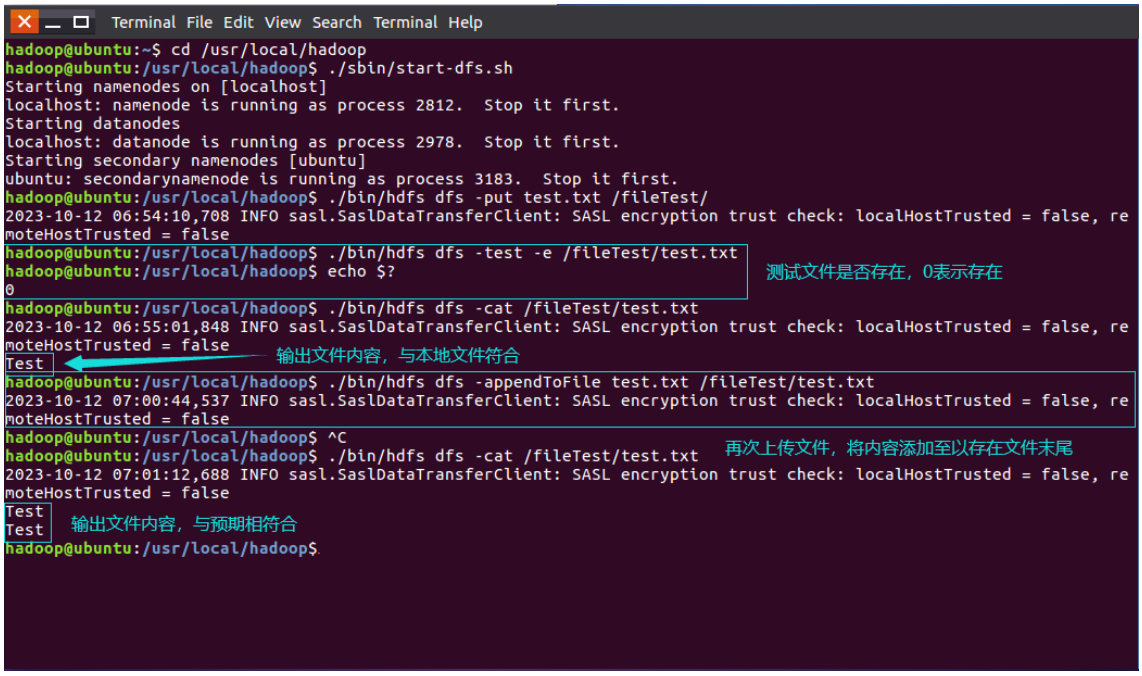
(1) 向 HDFS 中上传任意文本文件，如果指定的文件在 HDFS 中已经存在，由用户指定是追加到原有文件末尾还是覆盖原有的文件；

①Shell 命令：

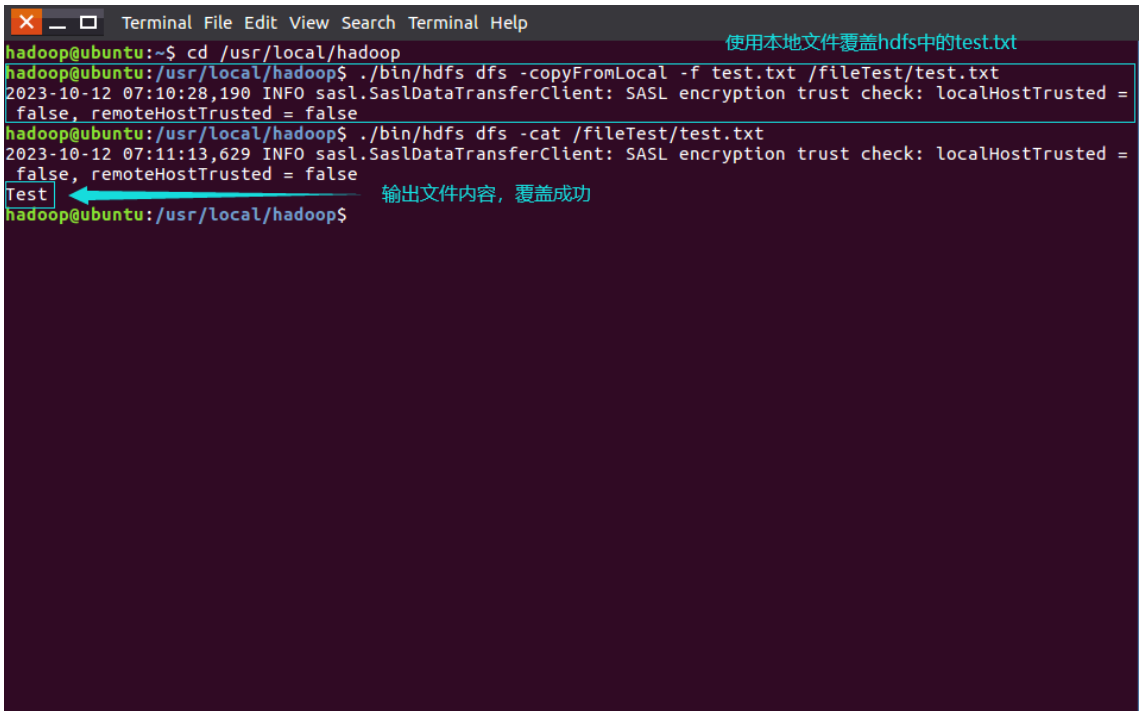
a. 本地 test 文本内容



b. 上传 test，检测文件是否存在，显示文件内容



c. 使用本地文件覆盖 HDFS 文件



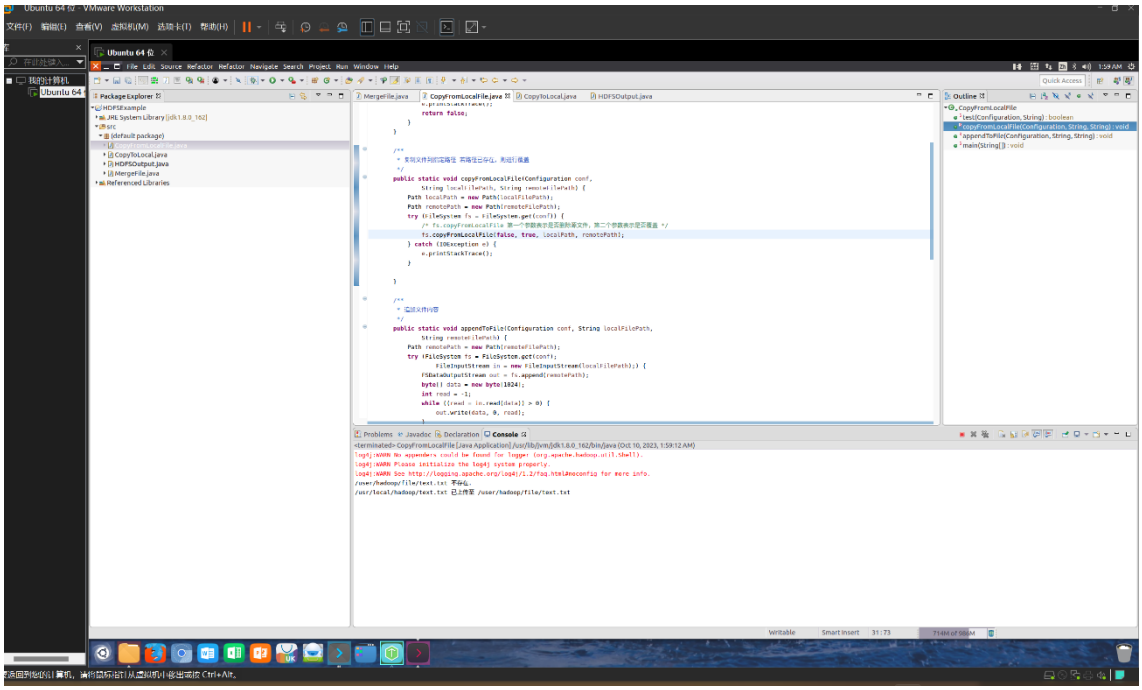
②Java：

源代码：

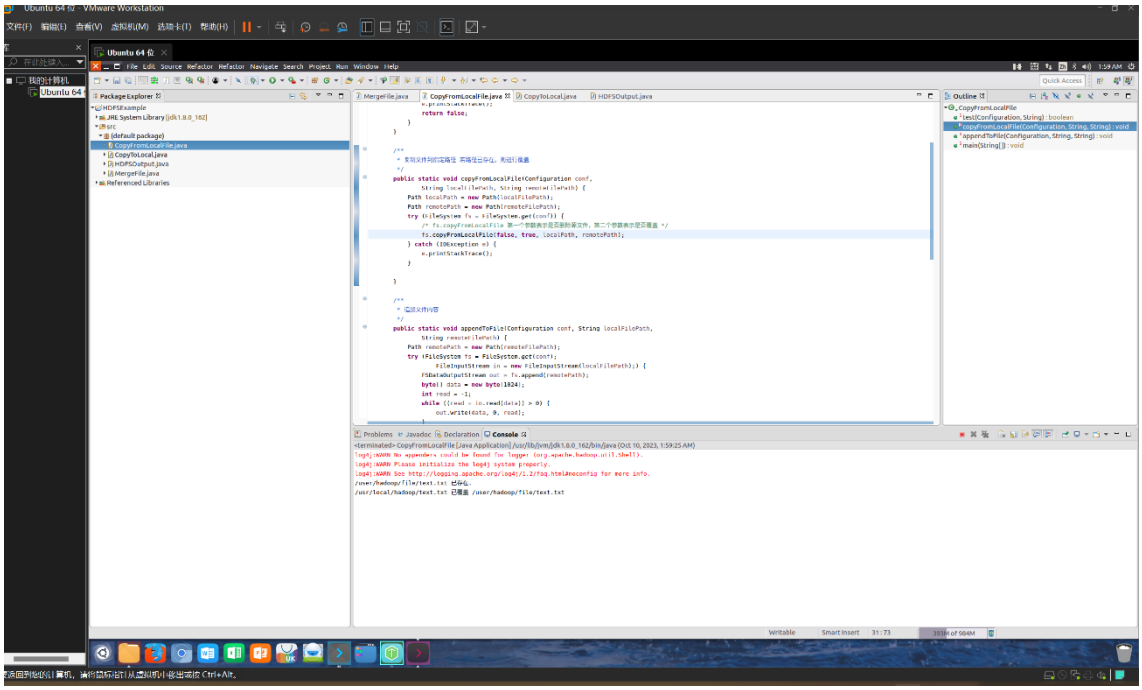
```java
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
public class CopyFromLocalFile {
    /**
     * 判断路径是否存在
     */
    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/hadoop/file/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 (CopyFromLocalFile.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
                CopyFromLocalFile.copyFromLocalFile(conf, localFilePath,
                        remoteFilePath);
                System.out.println(localFilePath + " 已上传至 " + remoteFilePath);
            } else if (choice.equals("overwrite")) { // 选择覆盖
                CopyFromLocalFile.copyFromLocalFile(conf, localFilePath,
                        remoteFilePath);
                System.out.println(localFilePath + " 已覆盖 " + remoteFilePath);
            } else if (choice.equals("append")) { // 选择追加
                CopyFromLocalFile.appendToFile(conf, localFilePath,
                        remoteFilePath);
                System.out.println(localFilePath + " 已追加至 " + remoteFilePath);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

实现结果：

a. 当文件在 HDFS 中不存在时



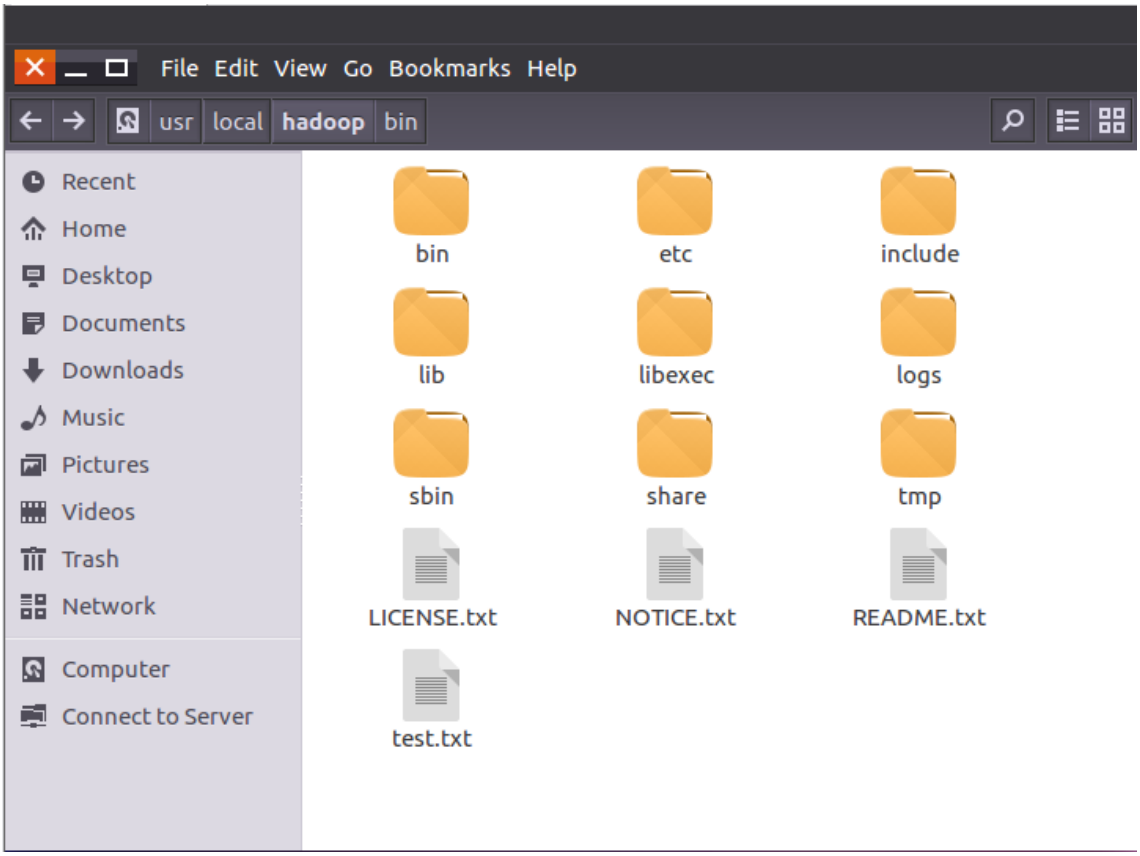
b. 当文件在 HDFS 中已经存在时



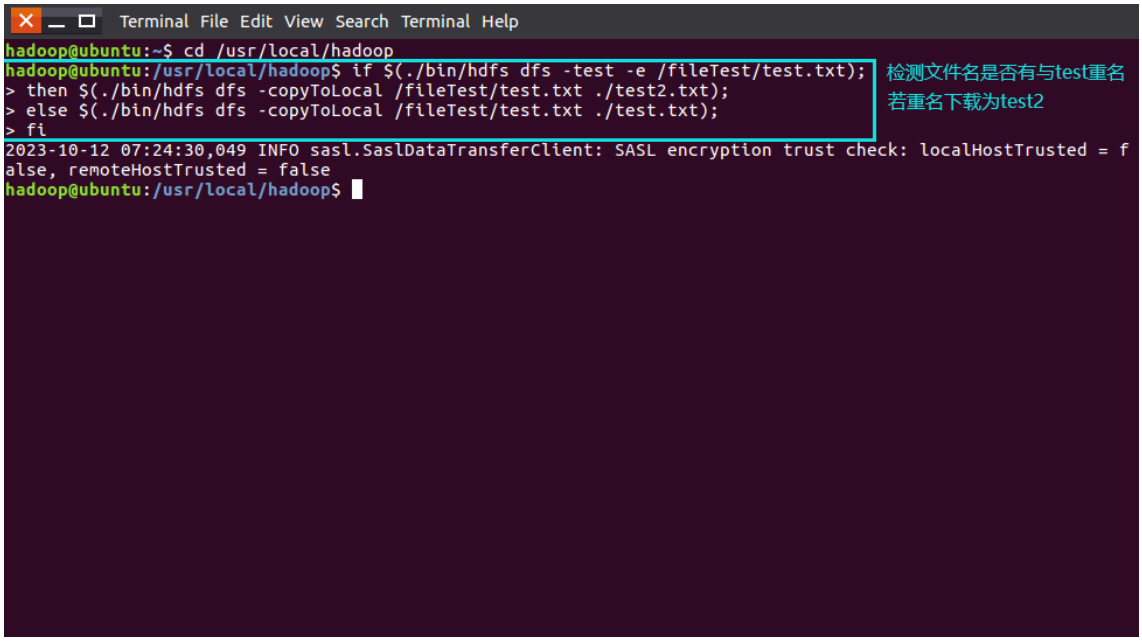
(2) 从 HDFS 中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名；

①Shell 命令：

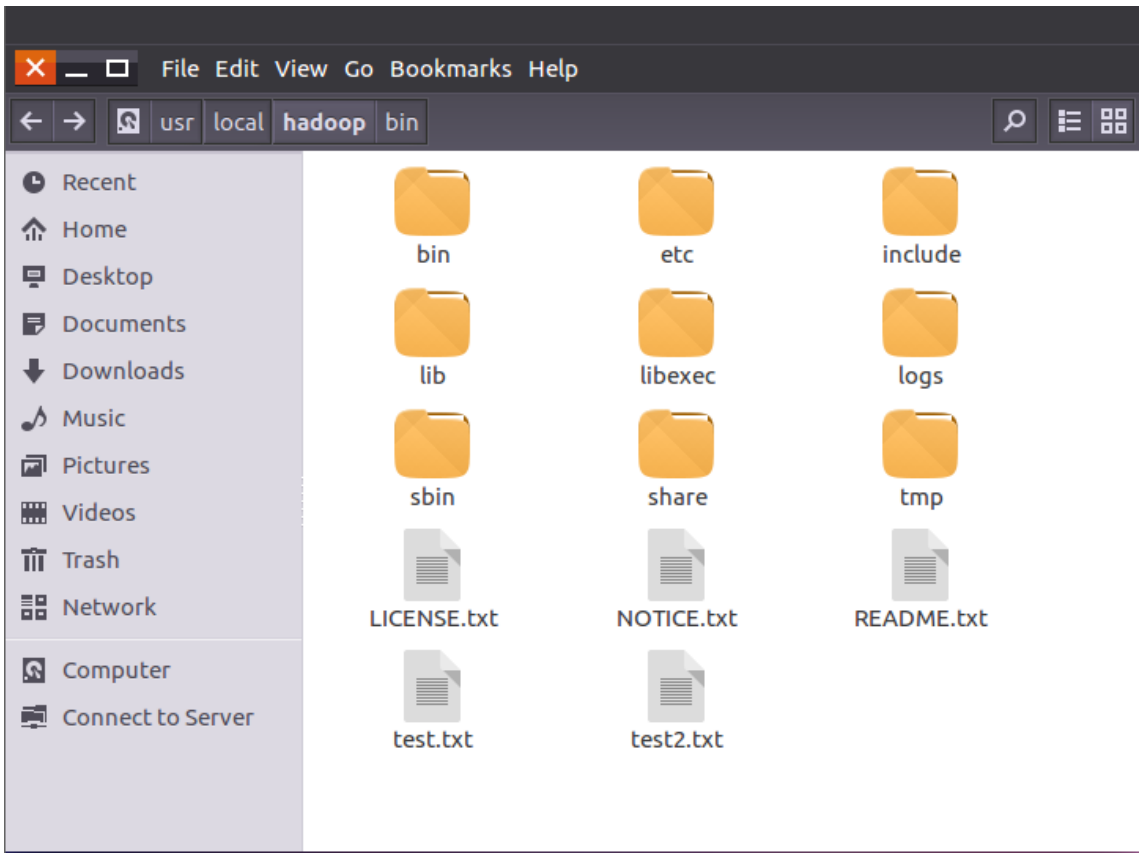
a. 操作前的文件目录



b. 文件下载，若与 test 重名，则重命名为 test2



c. 操作后的文件目录

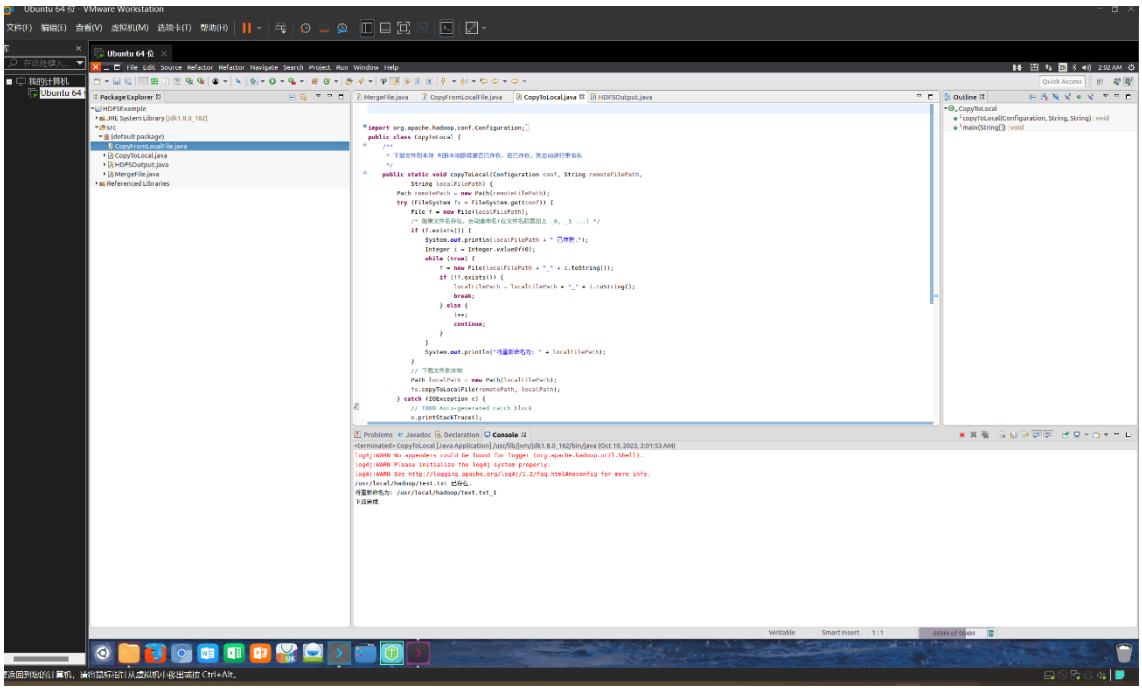


②Java：

源代码：

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

import java.io.*;
public class CopyToLocal {
    /**
     * 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) {
        Path remotePath = new Path(remoteFilePath);
        try (FileSystem fs = FileSystem.get(conf)) {
            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 = Integer.valueOf(0);
                while (true) {
                    f = new File(localFilePath + "_" + i.toString());
                    if (!f.exists()) {
                        localFilePath = localFilePath + "_" + i.toString();
                        break;
                    } else {
                        i++;
                        continue;
                    }
                }
                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);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            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/hadoop/file/text.txt"; // HDFS路径

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

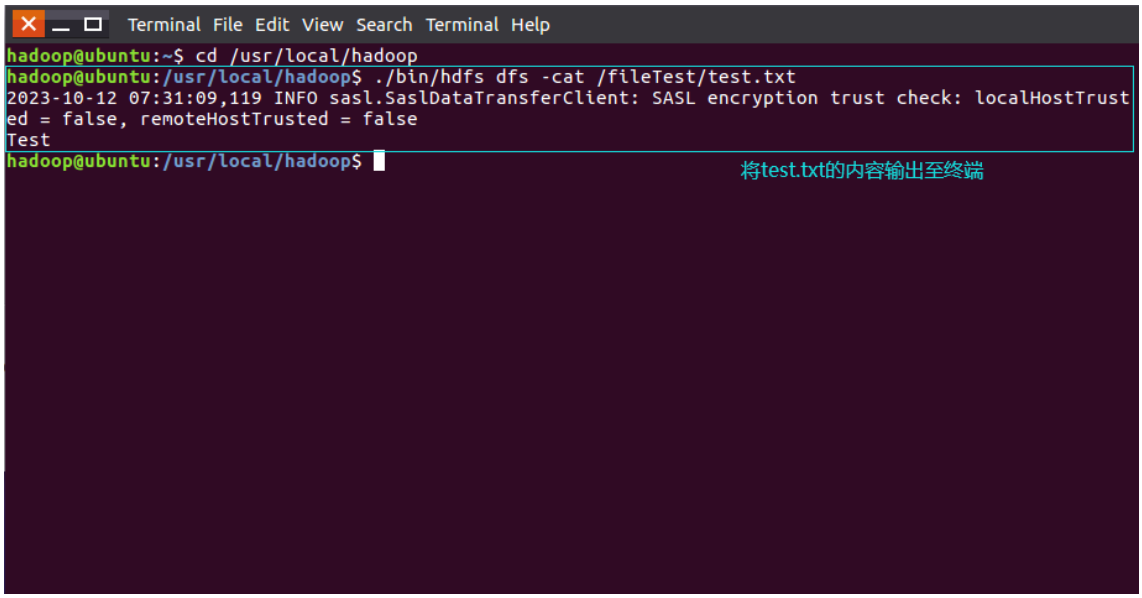
实现结果：



(3) 将 HDFS 中指定文件的内容输出到终端中；

①Shell 命令：

将 test.txt 的内容输出至终端

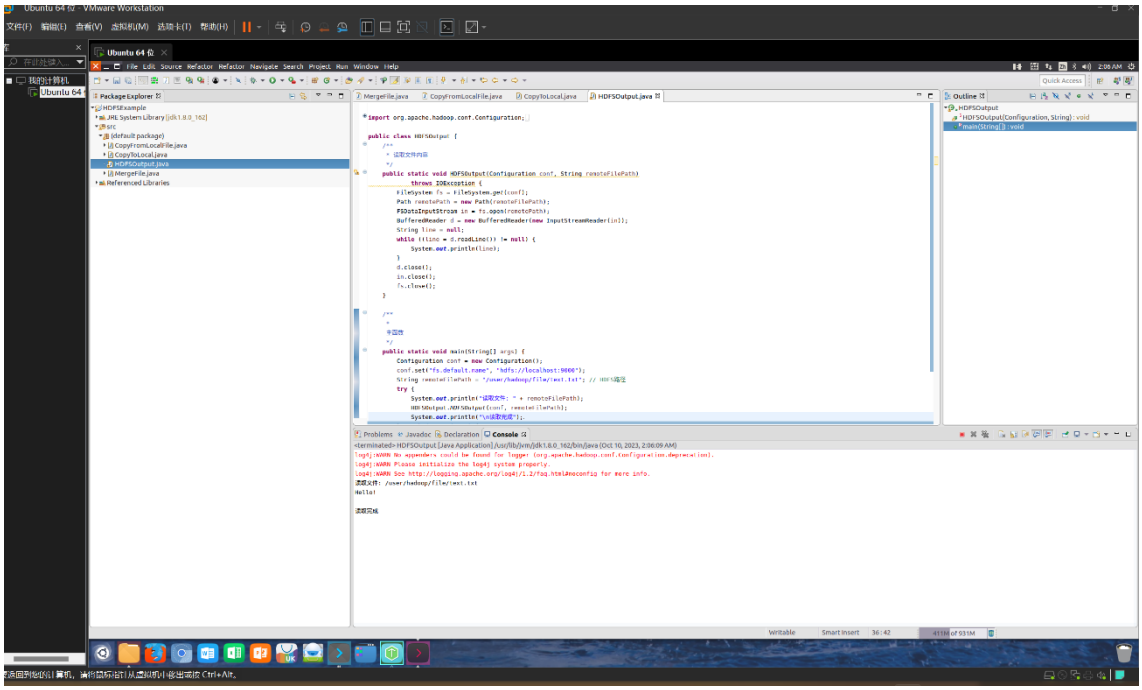


②Java：

源代码：

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

public class HDFSOutput {
    /**
     * 读取文件内容
     */
    public static void HDFSOutput(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();
    }
```

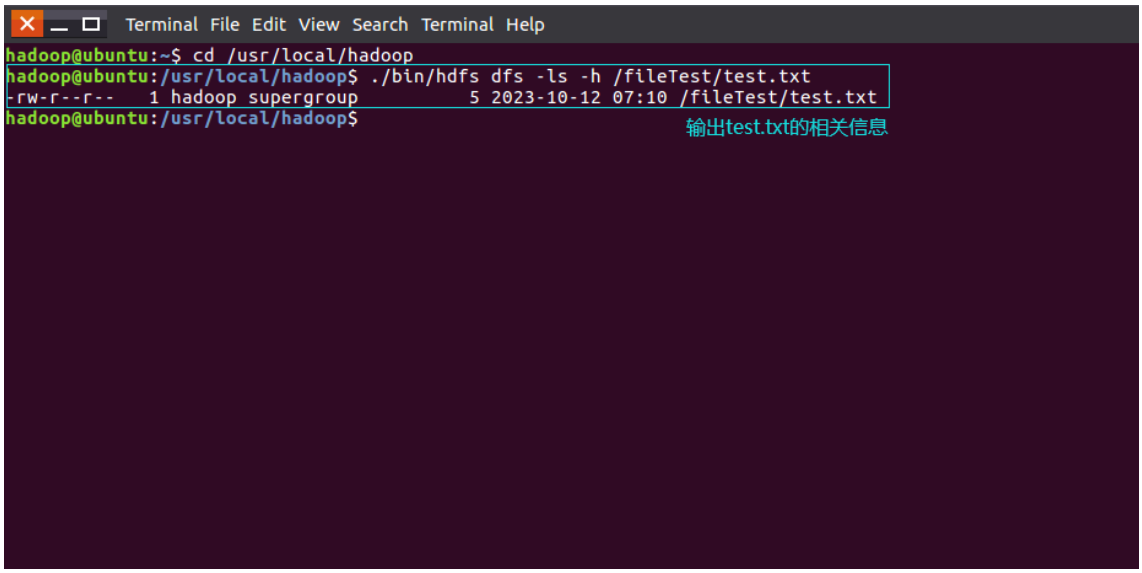
实现结果：



(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；

①Shell 命令：

输出 test.txt 的相关信息



②Java：

源代码：

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

public class HDFSReadAndShowDetail {
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/file1.txt";
        try {
            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());

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

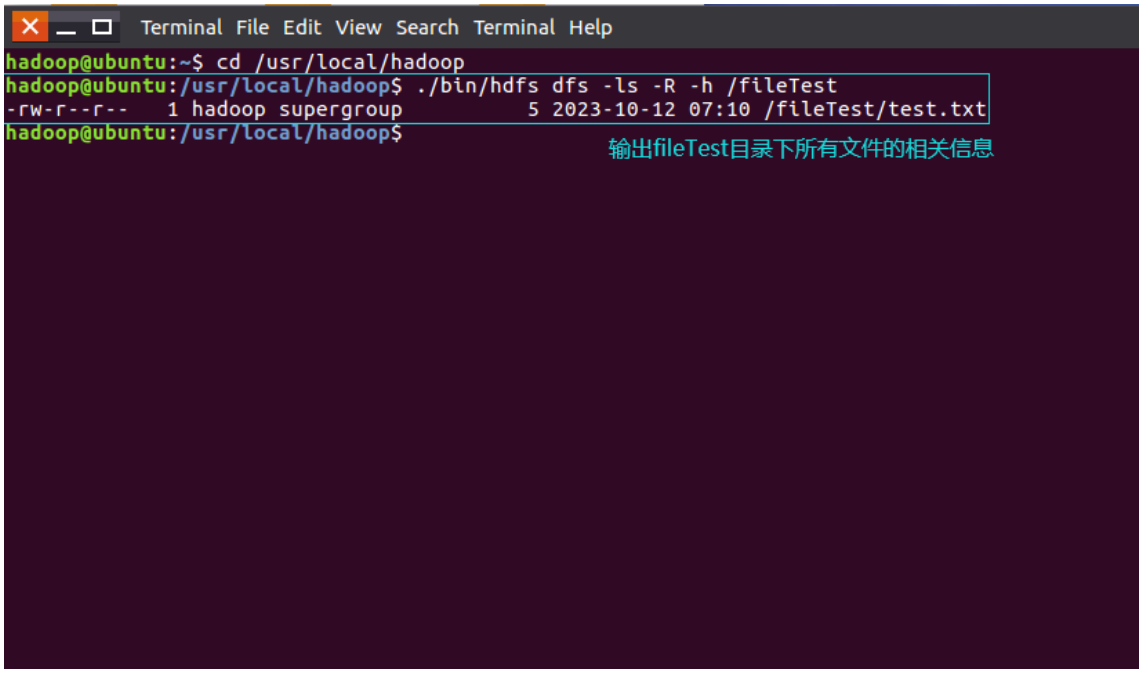
实现结果：

```
<terminated> HDFSReadAndShowDetail [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 3, 2023, 7:05:18
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
路径:hdfs://localhost:9000/user/hadoop/file1.txt
权限:rw-r--r--
大小:19
时间: 2023-09-22 20:47:24
```

(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；

①Shell 命令：

输出 fileTest 目录下的所有文件相关信息



②Java：

源代码：

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

public class HDFSFileInformationReader {
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://localhost:9000");
        String remoteDir = "/user/hadoop";
        try {
            System.out.println("读目录: " + remoteDir);

            FileSystem fs = FileSystem.get(conf);
            Path dirPath = new Path(remoteDir);
            RemoteIterator<LocatedFileStatus> remoteIterator = fs.listFiles(dirPath, true);
            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();
            System.out.println("读取完成");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

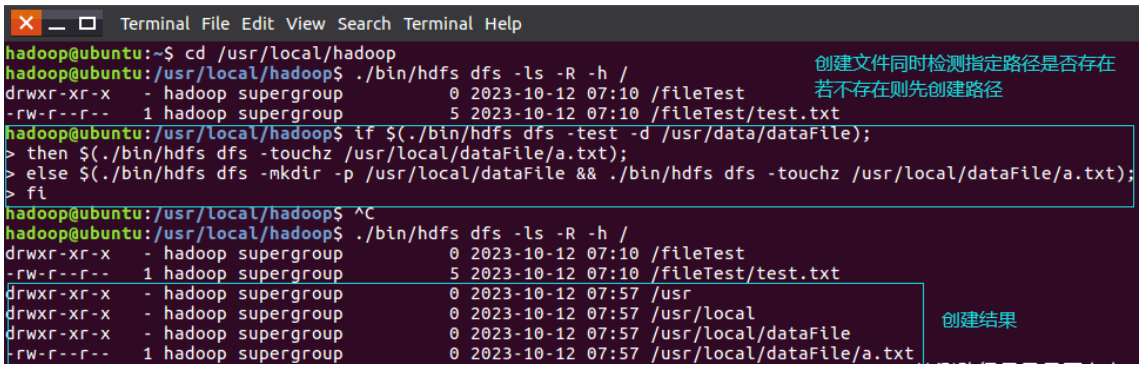
实现结果：

```
<terminated> HDFSFileInformationReader [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 3, 2023, 7:05:46 PM)
路径:hdfs://localhost:9000/user/hadoop/file1.txt
权限:rw-r--r--
大小:19
时间:2023-09-22 20:47:24

路径:hdfs://localhost:9000/user/hadoop/file2.txt
权限:rw-r--r--
大小:19
时间:2023-09-22 20:47:24

路径:hdfs://localhost:9000/user/hadoop/file3.txt
权限:rw-r--r--
大小:19
时间:2023-09-22 20:47:24
```

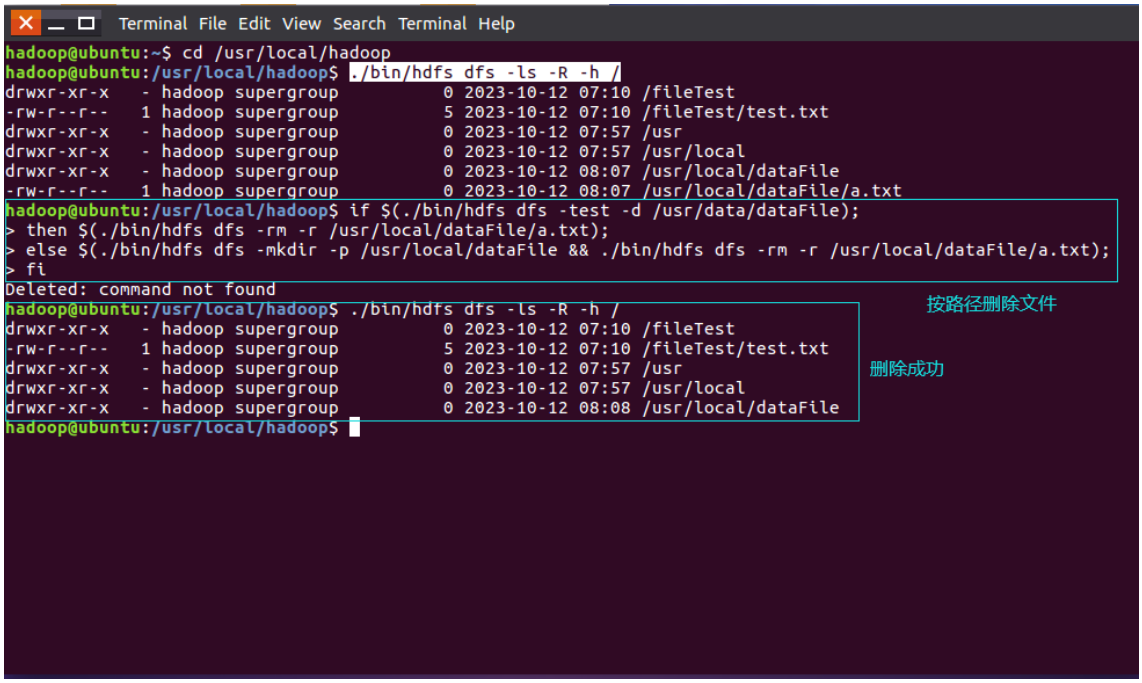
(6) 提供一个 HDFS 内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录；

①Shell 命令：

a. 按路径创建文件，若路径不存在先创建路径



b. 按路径删除文件



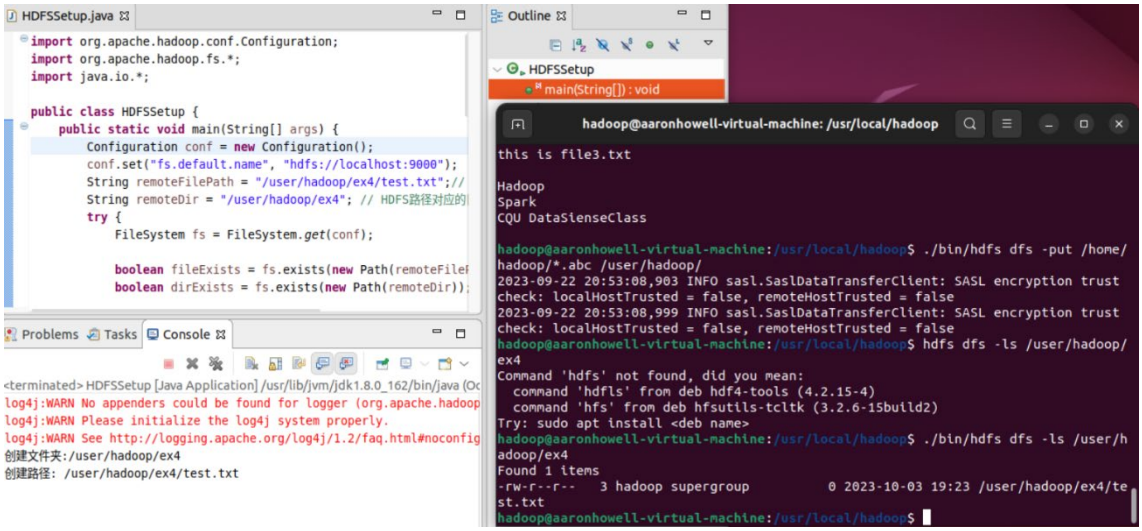
②Java：

源代码：

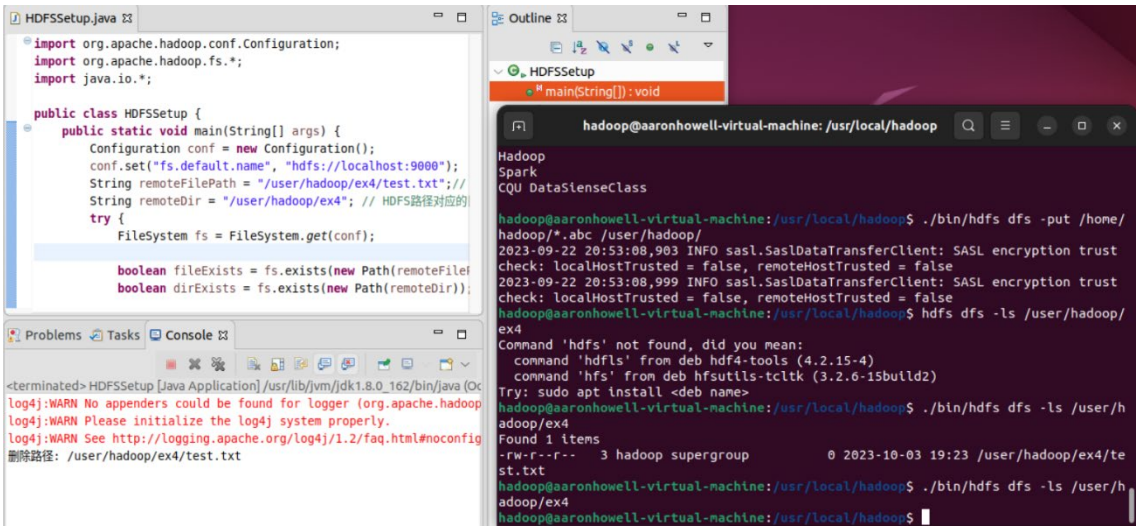
```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;

public class HDFSSetup {
    public static void main(String[] args) {
        Configuration conf = new Configuration();
        conf.set("fs.default.name", "hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/ex4/test.txt";// HDFS路径
        String remoteDir = "/user/hadoop/ex4"; // HDFS路径对应的目录
        try {
            FileSystem fs = FileSystem.get(conf);

            boolean fileExists = fs.exists(new Path(remoteFilePath));
            boolean dirExists = fs.exists(new Path(remoteDir));

            if (fileExists) {
                deleteFile(fs, remoteFilePath);
                System.out.println("删除路径: " + remoteFilePath);
            } else {
                if (!dirExists) {
                    createDirectory(fs, remoteDir);
                    System.out.println("创建文件夹:" + remoteDir);
                    createFile(fs, remoteFilePath);
                    System.out.println("创建路径: " + remoteFilePath);
                }
            }

            fs.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void deleteFile(FileSystem fs, String remoteFilePath) throws IOException
        Path remotePath = new Path(remoteFilePath);
        fs.delete(remotePath, false);
    }

    private static void createDirectory(FileSystem fs, String remoteDir) throws IOException
        Path dirPath = new Path(remoteDir);
        fs.mkdirs(dirPath);
    }

    private static void createFile(FileSystem fs, String remoteFilePath) throws IOException
        Path remotePath = new Path(remoteFilePath);
        FSDataOutputStream outputStream = fs.create(remotePath);
        outputStream.close();
    }
}
```

实现结果：

a. 原本目录不存在，我们运行程序，程序提示已经创建，用 shell 命令进行验证，可以发现，文件确已创建；



b. 再次运行程序，此时程序应该删除指定文件；

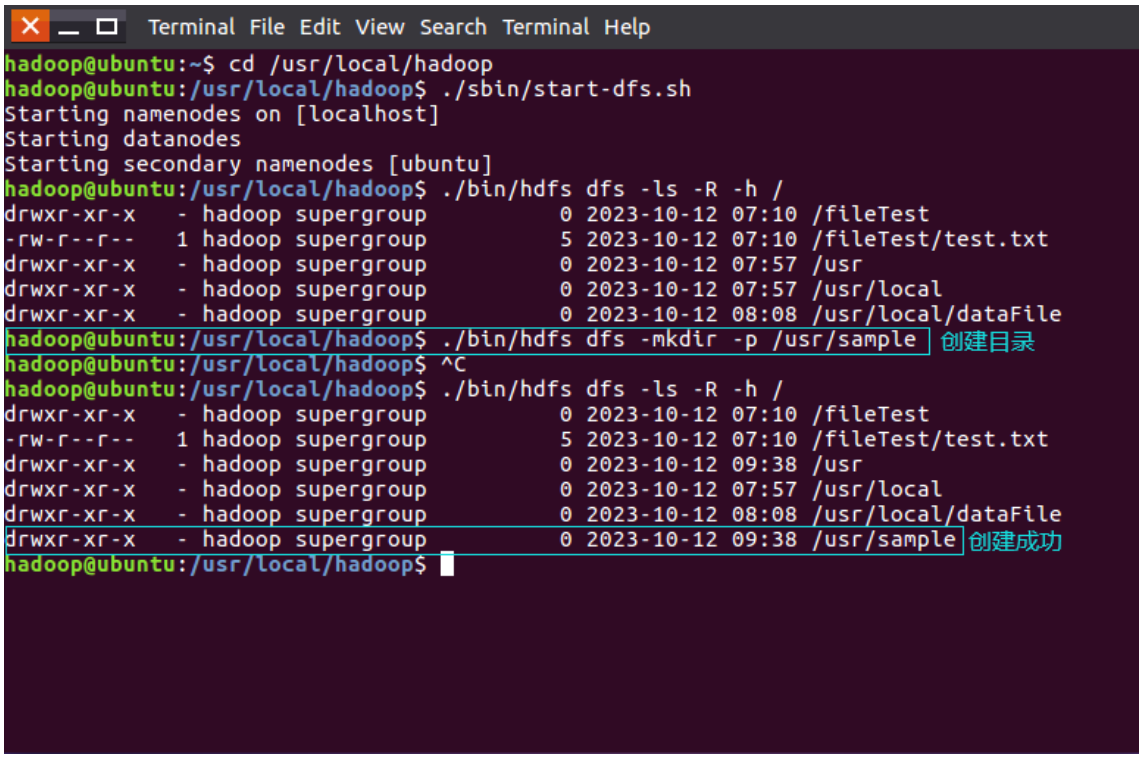


程序提示删除，通过 shell 命令，可知该目录下没有任何东西，确已删除。

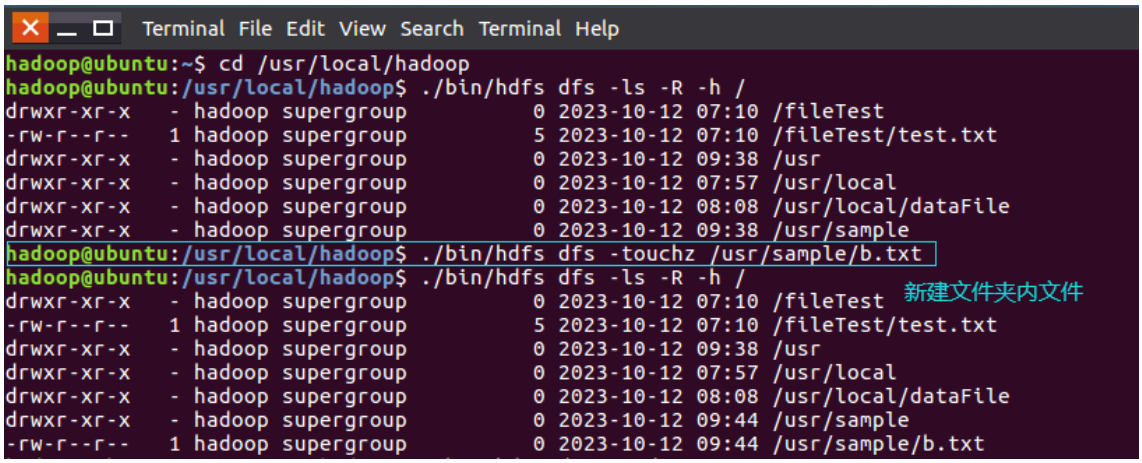
(7) 提供一个 HDFS 的目录的路径，对该目录进行创建和删除操作。创建目录时， 如果目录文件所在目录不存在则自动创建相应目录；删除目录时，由用户指定当该目录不为空时是否还删除该目录；

①Shell 命令：

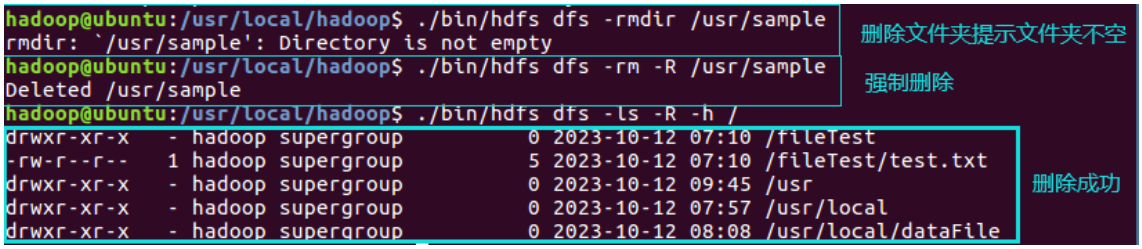
a. 按路径创建目录



b. 新建目录文件



c. 测试删除目录，提示有文件，测试强制删除，删除成功



②Java：

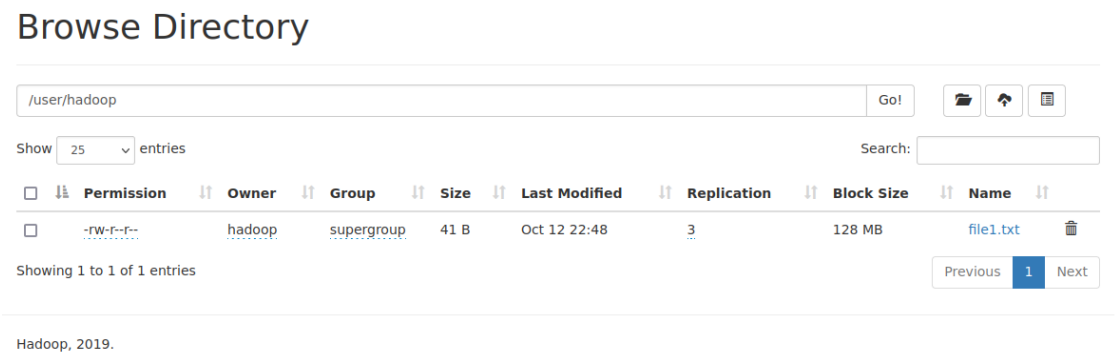
（方案一）

源代码：

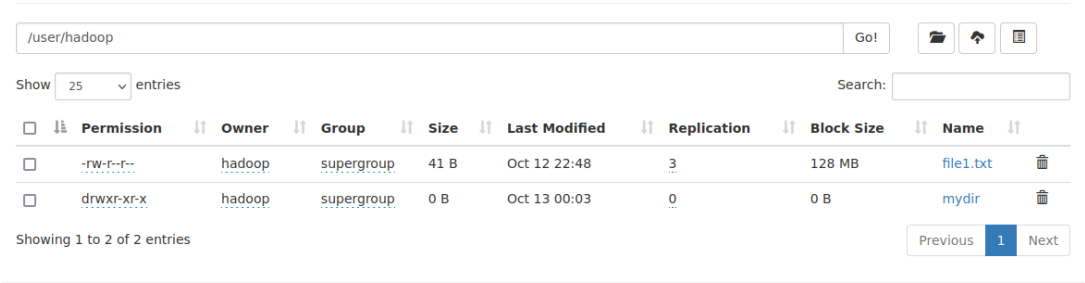
```java
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
public class HdfsDirectoryManager {

    private FileSystem fs;

    public HdfsDirectoryManager() throws IOException {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS","hdfs://localhost:9000");
        conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
        fs = FileSystem.get(conf);
    }

    public void createDirectory(String path) throws IOException {
        Path hdfsPath = new Path(path);
        if (!fs.exists(hdfsPath)) {
            fs.mkdirs(hdfsPath);
            System.out.println("Directory created: " + path);
        } else {
            System.out.println("Directory already exists: " + path);
        }
    }

    public void deleteDirectory(String path, boolean recursive) throws IOException {
        Path hdfsPath = new Path(path);
        if (fs.exists(hdfsPath)) {
            fs.delete(hdfsPath, recursive);
            System.out.println("Directory deleted: " + path);
        } else {
            System.out.println("Directory does not exist: " + path);
        }
    }

    public static void main(String[] args) throws IOException {
        HdfsDirectoryManager manager = new HdfsDirectoryManager();

        // 创建目录
        String directoryPath = "/user/hadoop/mydir";
        manager.createDirectory(directoryPath);

//        // 删除目录
//        boolean recursive = true;  // 如果目录不为空，也删除该目录
//        manager.deleteDirectory(directoryPath, recursive);

    }
}
```

实现结果：

a. 创建前



b. 创建后



（方案二）

源代码：

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

public class HDFSFileMakeOrRemove {

    public static void main(String[] args) {
        Configuration hdfsConfiguration = new Configuration();
        hdfsConfiguration.set("fs.default.name", "hdfs://localhost:9000");
        String hdfsDirectory = "/user/hadoop/ex4"; // HDFS 目录

        try (FileSystem fileSystem = FileSystem.get(hdfsConfiguration)) {
            if (!fileSystem.exists(new Path(hdfsDirectory)) || isHdfsDirectoryEmpty(fileSyste
                if (!fileSystem.exists(new Path(hdfsDirectory))) {
                    fileSystem.mkdirs(new Path(hdfsDirectory)); // 创建目录
                    System.out.println("创建目录: " + hdfsDirectory);
                } else {
                    removeHdfsDirectory(fileSystem, hdfsDirectory); // 删除目录
                    System.out.println("删除目录: " + hdfsDirectory);
                }
            } else {
                System.out.println("目录不为空，不删除: " + hdfsDirectory);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static boolean isHdfsDirectoryEmpty(FileSystem fileSystem, String hdfsDirectory) t
        Path hdfsPath = new Path(hdfsDirectory);
        RemoteIterator<LocatedFileStatus> remoteIterator = fileSystem.listFiles(hdfsPath, tru
        return !remoteIterator.hasNext();
    }

    public static boolean removeHdfsDirectory(FileSystem fileSystem, String hdfsDirectory) th
        Path hdfsPath = new Path(hdfsDirectory);
        boolean result = fileSystem.delete(hdfsPath, true);
        return result;
    }

    public static boolean askUserToDeleteHdfsDirectory() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("目录不为空，是否删除 (Y/N)? ");
        String choice = scanner.nextLine().trim().toLowerCase();
        return choice.equals("y");
    }
}
```

实现结果：

a.

```
check: localHostTrusted = false, remoteHostTrusted = false
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ hdfs dfs -ls /user/hadoop/
ex4
Command 'hdfs' not found, did you mean:
  command 'hdfls' from deb hdf4-tools (4.2.15-4)
  command 'hfs' from deb hfsutils-tcltk (3.2.6-15build2)
Try: sudo apt install <deb name>
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
Found 1 items
-rw-r--r--   3 hadoop supergroup          0 2023-10-03 19:23 /user/hadoop/ex4/te
st.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -touchz /us
er/hadoop/ex4/test.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
Found 1 items
-rw-r--r--   1 hadoop supergroup          0 2023-10-12 18:17 /user/hadoop/ex4/te
st.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$
```

b. 目录下创建一个 test.txt 然后运行程序，可选择删除或是不删除

```
目录不为空，是否删除 (Y/N)? N
目录不为空，不删除: /user/hadoop/ex4
```

```
目录不为空，是否删除 (Y/N)? Y
删除目录: /user/hadoop/ex4
```

c. 选择删除后，确认已删除

```
Command 'hdfs' not found, did you mean:
  command 'hdfls' from deb hdf4-tools (4.2.15-4)
  command 'hfs' from deb hfsutils-tcltk (3.2.6-15build2)
Try: sudo apt install <deb name>
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
Found 1 items
-rw-r--r--   3 hadoop supergroup          0 2023-10-03 19:23 /user/hadoop/ex4/te
st.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -touchz /us
er/hadoop/ex4/test.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
Found 1 items
-rw-r--r--   1 hadoop supergroup          0 2023-10-12 18:17 /user/hadoop/ex4/te
st.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
ls: `/user/hadoop/ex4': No such file or directory
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$
```

d. 再次运行代码，成功创建目录

```
log4j:WARN No appenders could be found for lo
log4j:WARN Please initialize the log4j syster
log4j:WARN See http://logging.apache.org/log4
创建目录: /user/hadoop/ex4
```

```
  command 'hfs' from deb hfsutils-tcltk (3.2.6-15build2)
Try: sudo apt install <deb name>
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
Found 1 items
-rw-r--r--   3 hadoop supergroup          0 2023-10-03 19:23 /user/hadoop/ex4/te
st.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -touchz /us
er/hadoop/ex4/test.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
Found 1 items
-rw-r--r--   1 hadoop supergroup          0 2023-10-12 18:17 /user/hadoop/ex4/te
st.txt
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
ls: `/user/hadoop/ex4': No such file or directory
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/h
adoop/ex4
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$
```

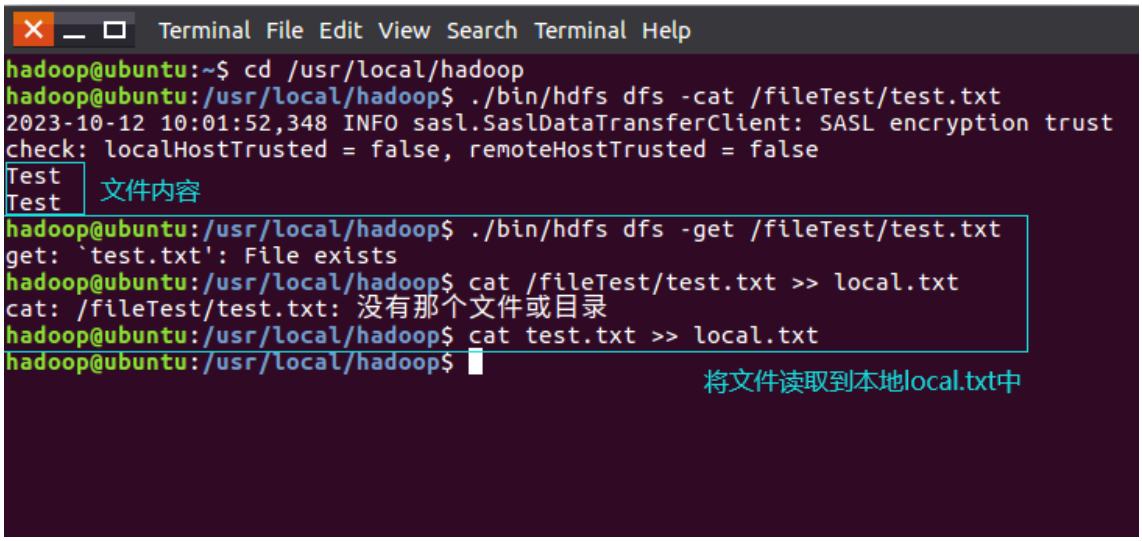
(8) 向 HDFS 中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾；

①Shell 命令：

a. 向指定文件末尾添加指定内容

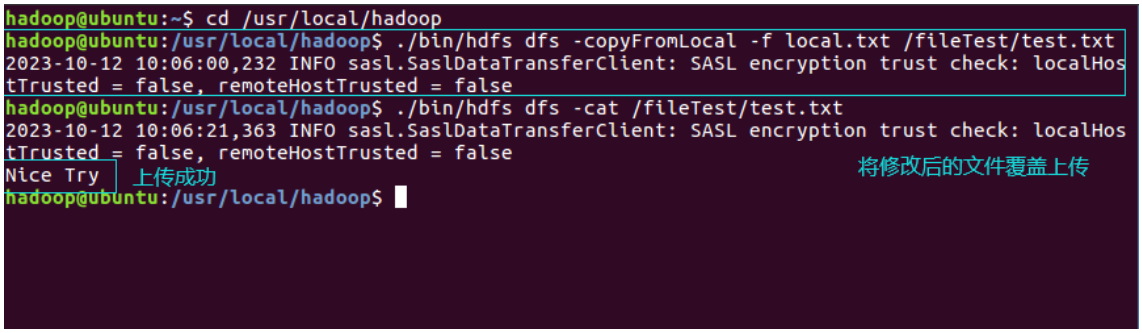
```
Terminal File Edit View Search Terminal Help
hadoop@ubuntu:~$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -cat /fileTest/test.txt
2023-10-12 09:53:24,109 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
 = false, remoteHostTrusted = false
Test   原内容
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -appendToFile test.txt /fileTest/test.txt
2023-10-12 09:53:36,888 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
 = false, remoteHostTrusted = false
hadoop@ubuntu:/usr/local/hadoop$ ^C
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -cat /fileTest/test.txt   向指定文件添加指定内容
2023-10-12 09:53:51,241 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
 = false, remoteHostTrusted = false
Test
Test   添加成功
hadoop@ubuntu:/usr/local/hadoop$
```

b. 将 HDFS 文件下载至本地



c. 修改下载的本地内容

d. 上传并覆盖



②Java：

（方案一）

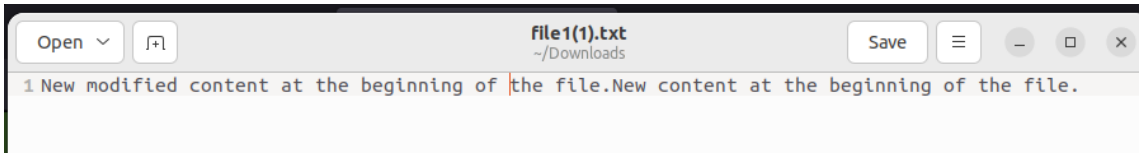
源代码：

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

import java.io.ByteArrayOutputStream;
import java.io.IOException;

public class HdfsFileManager {

    private FileSystem fs;

    public HdfsFileManager() throws IOException {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS","hdfs://localhost:9000");
        conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
        fs = FileSystem.get(conf);
    }

    public void appendToFile(String path, String content, boolean appendToEnd) throws IOExcep
        Path hdfsPath = new Path(path);
        if (!fs.exists(hdfsPath)) {
            System.out.println("File does not exist: " + path);
            return;
        }

        FSDataOutputStream outputStream;
        if (appendToEnd) {
            outputStream = fs.append(hdfsPath);
            outputStream.writeBytes(content);
        } else {
            // Hadoop does not support prepending to a file directly.
            // You need to read the original content, then write the new content and the orig
            FSDataInputStream inputStream = fs.open(hdfsPath);
            ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
            byte[] buffer = new byte[1024];
            int length;
            while ((length = inputStream.read(buffer)) != -1) {
                byteArrayOutputStream.write(buffer, 0, length);
            }
            String originalContent = byteArrayOutputStream.toString();
            inputStream.close();

            outputStream = fs.create(hdfsPath, true);  // overwrite the file
            outputStream.writeBytes(content + originalContent);
        }
        outputStream.close();
    }

    public static void main(String[] args) throws IOException {
        HdfsFileManager manager = new HdfsFileManager();

//        // 追加内容到文件的末尾
//        String filePath = "/user/hadoop/file1.txt";
//        manager.appendToFile(filePath, "New content at the end of the file.", true);

//         追加内容到文件的开头
        String filePath = "/user/hadoop/file1.txt";
        manager.appendToFile(filePath, "New modified content at the beginning of the file.",
    }
}
```

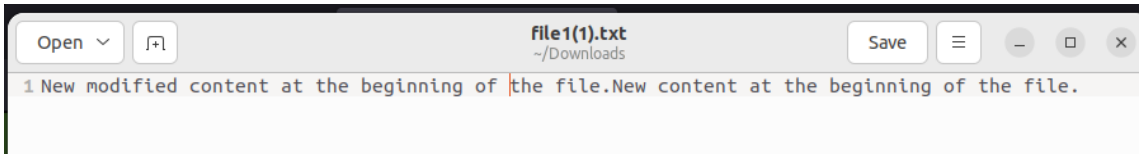
实现结果：

a. 原始文件

b. 增加在末尾



c. 增加在开头



（方案二）

源代码：

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.OutputStreamWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class HDFSFileAppend {

    public static void main(String[] args) {
        Configuration hdfsConfiguration = new Configuration();
        hdfsConfiguration.set("fs.default.name", "hdfs://localhost:9000");
        String hdfsFilePath = "/user/hadoop/ex4/test.txt"; // HDFS 文件路径

        try (FileSystem fileSystem = FileSystem.get(hdfsConfiguration)) {
            if (fileSystem.exists(new Path(hdfsFilePath)) && appendContentToFile(fileSystem,
                System.out.println("内容已追加到文件: " + hdfsFilePath);
            } else {
                System.out.println("文件不存在或操作取消: " + hdfsFilePath);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static boolean appendContentToFile(FileSystem fileSystem, String hdfsFilePath) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("是否追加到文件开头 (Y/N)? ");
        String choice = scanner.nextLine().trim().toLowerCase();

        if (choice.equals("y")) {
            return appendContent(fileSystem, hdfsFilePath, true);
        } else if (choice.equals("n")) {
            return appendContent(fileSystem, hdfsFilePath, false);
        } else {
            System.out.println("无效选择，操作取消。");
            return false;
        }
    }

    public static boolean appendContent(FileSystem fileSystem, String hdfsFilePath, boolean a
        try {
            String content = getUserInputContent();
            FSDataOutputStream outputStream = fileSystem.append(new Path(hdfsFilePath));
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outputStream));
            if (appendToBeginning) {
                writer.close();
                    String existingContent = "";
                    Path hdfsPath = new Path(hdfsFilePath);
                    FSDataInputStream inputStream = fileSystem.open(hdfsPath);
                    BufferedReader reader = new BufferedReader(new InputStreamReader(inputStr
                    String line;
                    while ((line = reader.readLine()) != null) {
                        existingContent += line + "\n";
                    }
                    reader.close();

                    // 合并原内容和新内容
                    content = content + existingContent;

                    // 清空源文件
                    FSDataOutputStream clearOutputStream = fileSystem.create(hdfsPath, true);
                    BufferedWriter clearWriter = new BufferedWriter(new OutputStreamWriter(cl
                    clearWriter.write("");
                    clearWriter.close();

                    // 写入合并的字符串
                    FSDataOutputStream outputStream1 = fileSystem.append(hdfsPath);
                    BufferedWriter writer1 = new BufferedWriter(new OutputStreamWriter(output
                    writer1.write(content);
                    writer1.close();

                } else {
                    writer.append(content);
                    writer.close();
                }

                return true;
            } catch (IOException e) {
                e.printStackTrace();
                return false;
            }
        }

    public static String getUserInputContent() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("请输入要追加的内容: ");
        return scanner.nextLine();
    }
}
```

实现结果：

a. 初始文件：test.txt 中，原本包含的文字为 something

b. 增加在开头

```
<terminated> HDFSFileAppend [Java Application] /us
log4j:WARN No appenders could be found for l
log4j:WARN Please initialize the log4j syste
log4j:WARN See http://logging.apache.org/log
是否追加到文件开头 (Y/N)? Y
请输入要追加的内容: Begin
内容已追加到文件: /user/hadoop/ex4/test.txt
```

```
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat /user/
hadoop/ex4/test.txt
2023-10-12 19:06:34,074 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localHostTrusted = false, remoteHostTrusted = false
BeginSomething
```

c. 增加在末尾

```
hadoop@aaronhowell-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat /user/
hadoop/ex4/test.txt
2023-10-12 19:38:07,000 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localHostTrusted = false, remoteHostTrusted = false
BeginSomethingEndinghadoop@aaronhowell-virtual-machine:/usr/local/hadoop$
```

(9) 删除 HDFS 中指定的文件；

①Shell 命令：

删除指定文件

```
Terminal File Edit View Search Terminal Help
hadoop@ubuntu:~$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -ls -R -h /
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 10:06 /fileTest
-rw-r--r--   1 hadoop supergroup          9 2023-10-12 10:06 /fileTest/test.txt
-rw-r--r--   1 hadoop supergroup          5 2023-10-12 09:52 /fileTest/text.txt
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 09:45 /usr
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 07:57 /usr/local
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 08:08 /usr/local/dataFile
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -rm -r /fileTest/text.txt
Deleted /fileTest/text.txt
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -ls -R -h /
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 10:09 /fileTest
-rw-r--r--   1 hadoop supergroup          9 2023-10-12 10:06 /fileTest/test.txt
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 09:45 /usr
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 07:57 /usr/local
drwxr-xr-x   - hadoop supergroup          0 2023-10-12 08:08 /usr/local/dataFile
hadoop@ubuntu:/usr/local/hadoop$
```

删除指定文件

删除成功

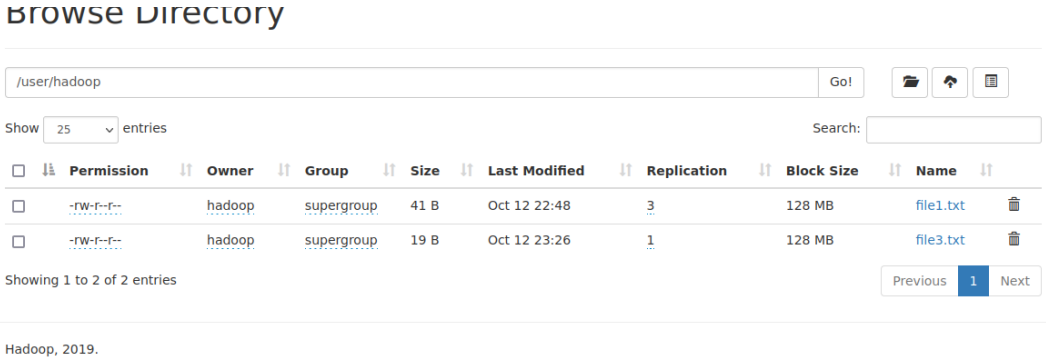
②Java：

源代码：

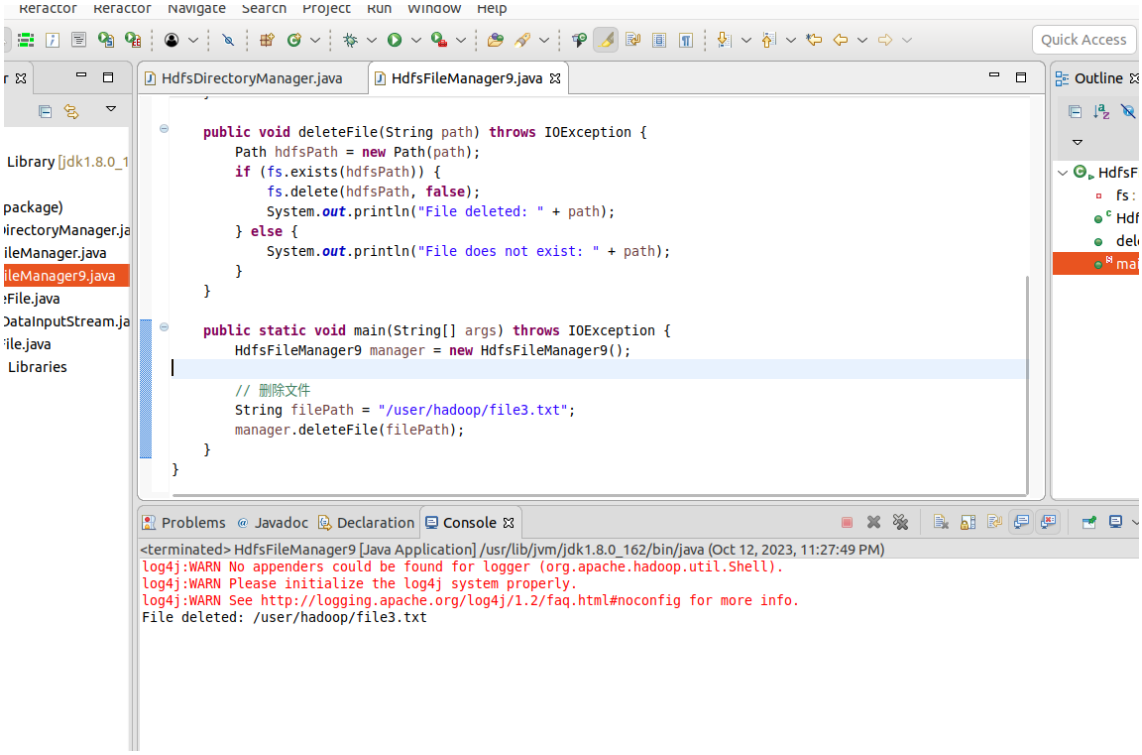
```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

import java.io.IOException;

public class HdfsFileManager9 {

    private FileSystem fs;

    public HdfsFileManager9() throws IOException {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS","hdfs://localhost:9000");
        conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
        fs = FileSystem.get(conf);
    }

    public void deleteFile(String path) throws IOException {
        Path hdfsPath = new Path(path);
        if (fs.exists(hdfsPath)) {
            fs.delete(hdfsPath, false);
            System.out.println("File deleted: " + path);
        } else {
            System.out.println("File does not exist: " + path);
        }
    }

    public static void main(String[] args) throws IOException {
        HdfsFileManager9 manager = new HdfsFileManager9();

        // 删除文件
        String filePath = "/user/hadoop/file3.txt";
        manager.deleteFile(filePath);
    }
}
```

实现效果：

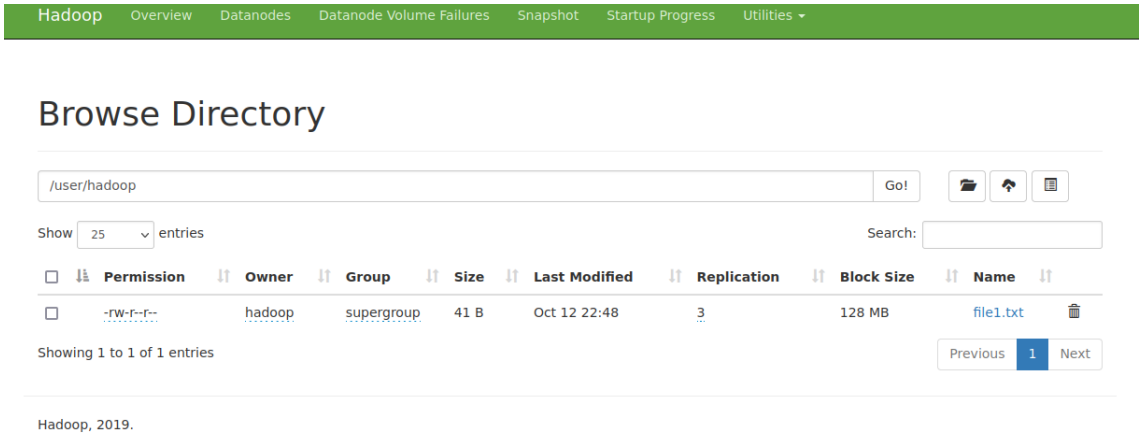
a. 删除前



b. 运行代码

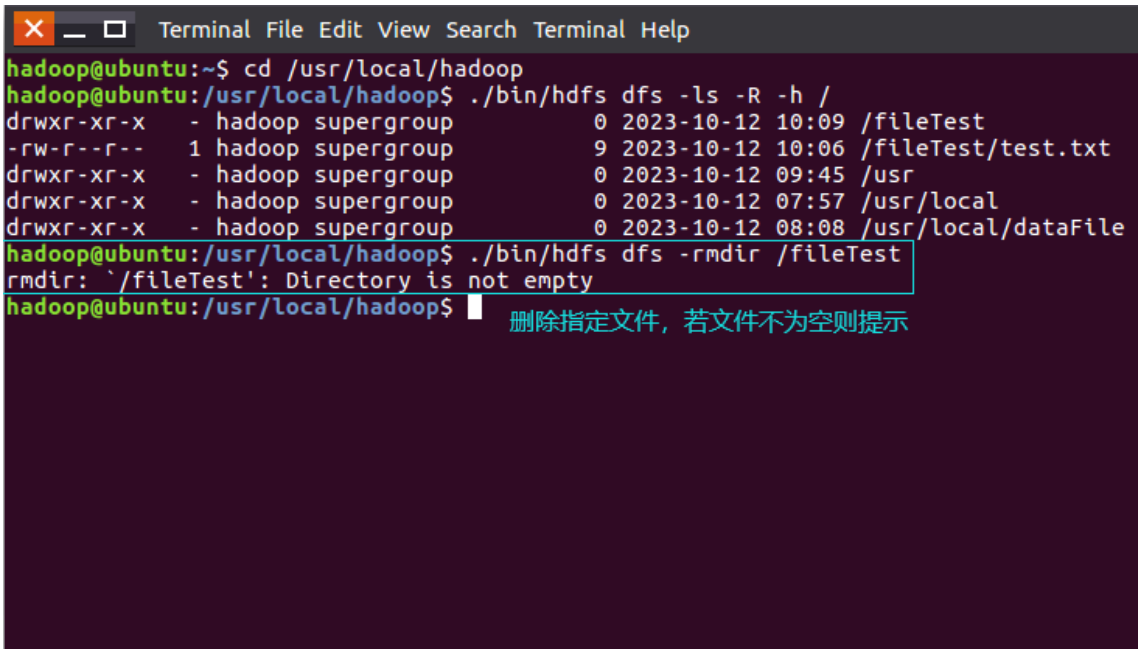

HdfsDirectoryManager.java
HdfsFileManager9.java
Outline
public void deleteFile(String path) throws IOException {
Path hdfsPath = new Path(path);
if (fs.exists(hdfsPath)) {
fs.delete(hdfsPath, false);
System.out.println("File deleted: " + path);
} else {
System.out.println("File does not exist: " + path);
}
}
public static void main(String[] args) throws IOException {
HdfsFileManager9 manager = new HdfsFileManager9();
// 删除文件
String filePath = "/user/hadoop/file3.txt";
manager.deleteFile(filePath);
}
}
Problems
Javadoc
Declaration
Console
<terminated> HdfsFileManager9 [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 12, 2023, 11:27:49 PM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
File deleted: /user/hadoop/file3.txt

c. 删除后


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/user/hadoop
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
hadoop
supergroup
41 B
Oct 12 22:48
3
128 MB
file1.txt
Showing 1 to 1 of 1 entries
Previous
1
Next
Hadoop, 2019.

(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；

①Shell 命令：
删除指定目录，不为空时提示



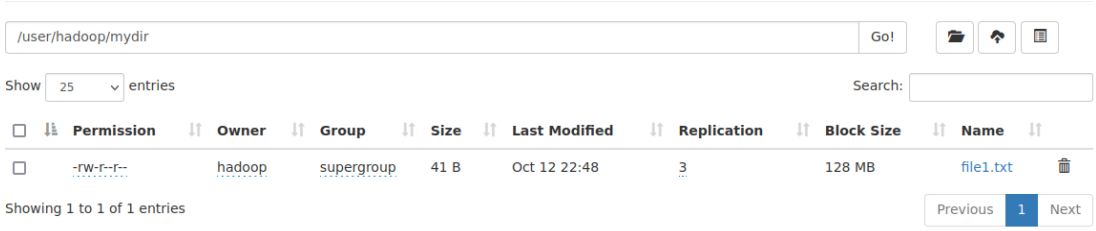
②Java：

（方案一）
源代码：

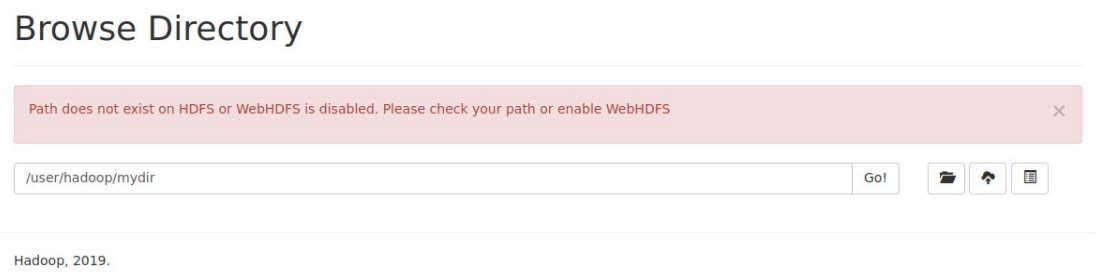
```java
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
public class HdfsDirectoryManager10 {

    private FileSystem fs;

    public HdfsDirectoryManager10() throws IOException {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS","hdfs://localhost:9000");
        conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
        fs = FileSystem.get(conf);
    }

    public void deleteDirectory(String path, boolean recursive) throws IOException {
        Path hdfsPath = new Path(path);
        if (fs.exists(hdfsPath)) {
            fs.delete(hdfsPath, recursive);
            System.out.println("Directory deleted: " + path);
        } else {
            System.out.println("Directory does not exist: " + path);
        }
    }

    public static void main(String[] args) throws IOException {
        HdfsDirectoryManager manager = new HdfsDirectoryManager();

        // 删除目录
        String directoryPath = "/user/hadoop/mydir";
        boolean recursive = true;  // 如果目录不为空，也删除该目录
        manager.deleteDirectory(directoryPath, recursive);
    }
}
```

运行结果：

删除前：



删除后：

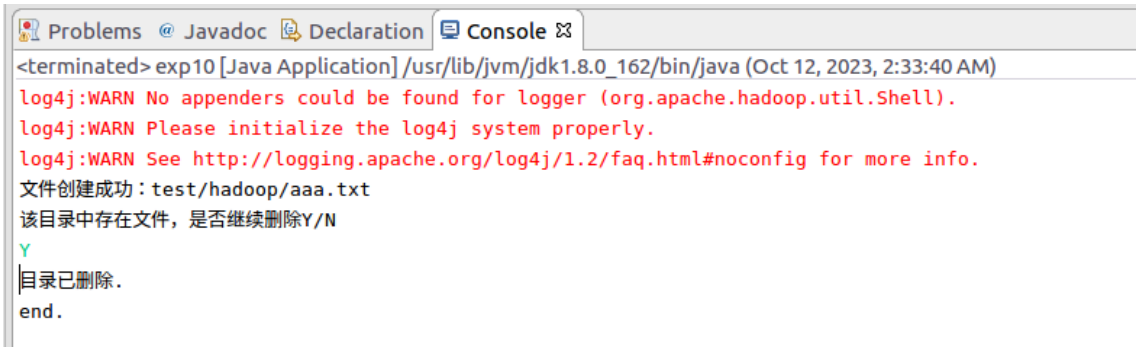


（方案二）

源代码：

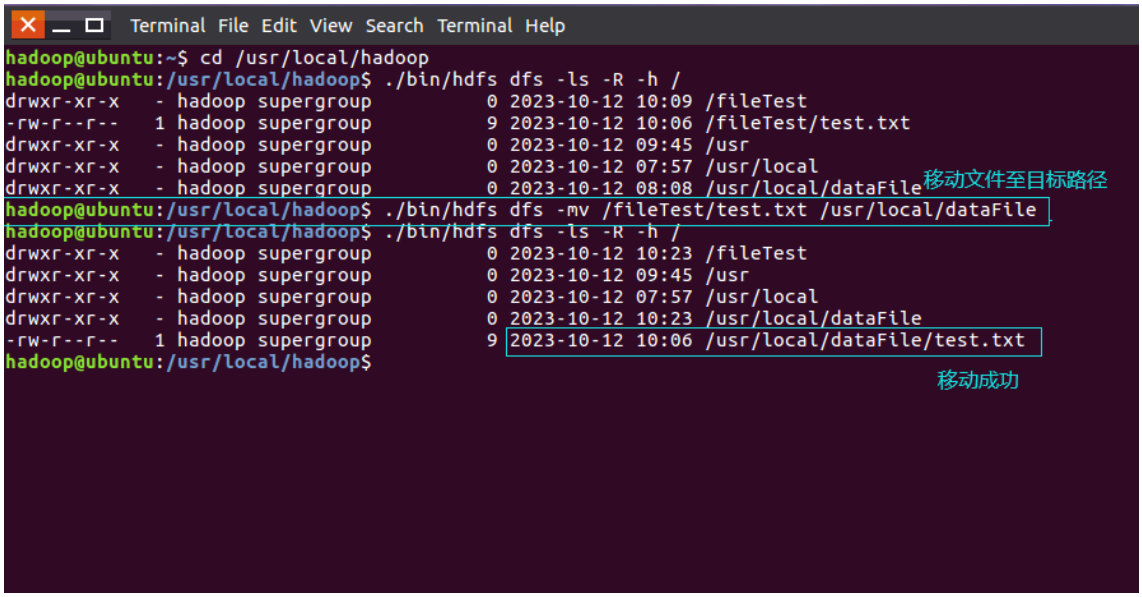
```
    public static void deleteDirectory(Configuration conf, String filePath) throws IOException {
        FileSystem fs = FileSystem.get(conf);
        Path path = new Path(filePath);
        FileStatus[] status = fs.listStatus(path);
        if(status.length > 0) {
            System.out.println("该目录中存在文件，是否继续删除Y/N");
            Scanner cin = new Scanner(System.in);
            String flag = cin.next();
            if(flag.equals("Y")) {
                fs.delete(path,true);
                System.out.println("目录已删除.");
            }else {
                System.out.println("用户取消删除.");
            }
        }else {
            fs.delete(path,true);
            System.out.println("目录已删除.");
        }
        fs.close();
    }

    public static void main(String[] args) throws IOException, InterruptedException, URISyntaxException {
        Scanner cin = new Scanner(System.in);
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://localhost:9000");
        conf.set("dfs.client.block.write.replace-datanode-on-failure.enable","true");
        conf.set("dfs.client.block.write.replace-datanode-on-failure.policy", "NEVER");
        conf.set("dfs.replication","1");
        String dirPath = "test/hadoop/aaa.txt";
        exp10.createFile(conf, dirPath);
        exp10.deleteDirectory(conf, "test");
        System.out.println("end.");
    }
}
```

实现结果：



(11) 在 HDFS 中，将文件从源路径移动到目的路径。

①Shell 命令：
移动指定文件至指定路径



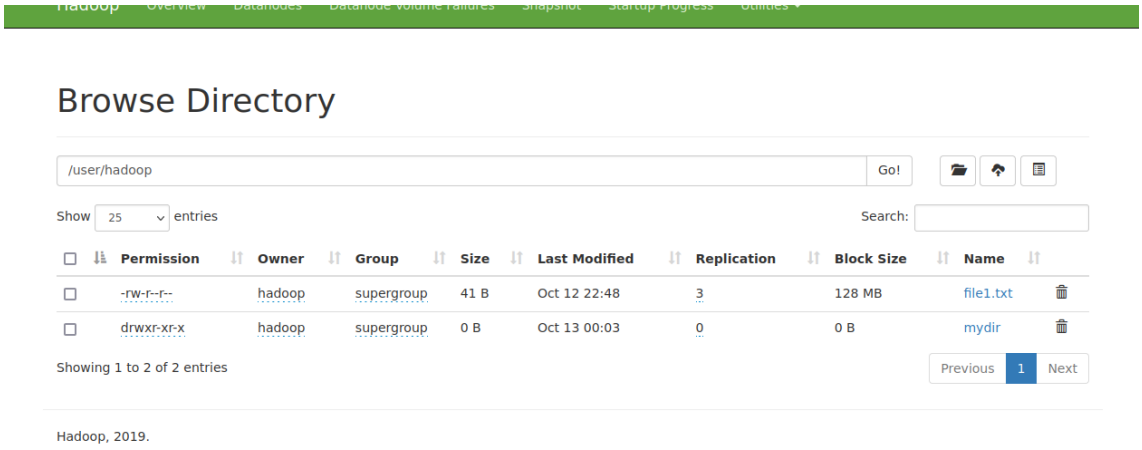
②Java：

（方案一）
源代码：

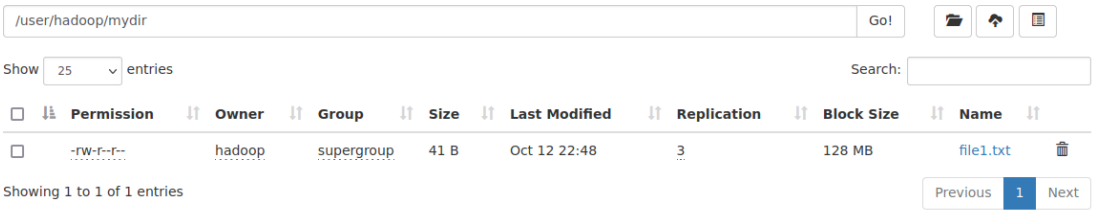
```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

import java.io.IOException;

public class HdfsFileManager11 {

    private FileSystem fs;

    public HdfsFileManager11() throws IOException {
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS","hdfs://localhost:9000");
        conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
        fs = FileSystem.get(conf);
    }

    public void moveFile(String sourcePath, String destinationPath) throws IOException {
        Path srcPath = new Path(sourcePath);
        Path dstPath = new Path(destinationPath);
        if (!fs.exists(srcPath)) {
            System.out.println("Source file does not exist: " + sourcePath);
            return;
        }
        if (fs.exists(dstPath)) {
            System.out.println("Destination file already exists: " + destinationPath);
            return;
        }
        boolean isMoved = fs.rename(srcPath, dstPath);
        if (isMoved) {
            System.out.println("File moved successfully from " + sourcePath + " to " + destin
        } else {
            System.out.println("Failed to move file from " + sourcePath + " to " + destinatio
        }
    }

    public static void main(String[] args) throws IOException {
        HdfsFileManager11 manager = new HdfsFileManager11();

        // 移动文件
        String sourceFilePath = "/user/hadoop/file1.txt";
        String destinationFilePath = "/user/hadoop/mydir/myfile.txt";
        manager.moveFile(sourceFilePath, destinationFilePath);
    }
}
```

实现结果：

a. 移动前



b. 移动后



（方案二）

源代码：

```
public static void moveFile(Configuration conf, String srcPath, String dirPath) throws IOException {
    FileSystem fs = FileSystem.get(conf);
    if(fs.exists(new Path(dirPath))) {
        System.out.println("文件被占用。");
        return;
    }
    if(fs.rename(new Path(srcPath), new Path(dirPath))) {
        System.out.println("文件移动成功。");
    } else {
        System.out.println("文件移动失败。");
    }
}
```

a. 文件移动失败

```
public static void main(String[] args) throws IOException, InterruptedException, URISyntaxException {
    Scanner cin = new Scanner(System.in);
    Configuration conf = new Configuration();
    conf.set("fs.defaultFS", "hdfs://localhost:9000");
    conf.set("dfs.client.block.write.replace-datanode-on-failure.enable","true");
    conf.set("dfs.client.block.write.replace-datanode-on-failure.policy", "NEVER");
    conf.set("dfs.replication","1");
    String filePath = "hadoop.txt";
    exp10.moveFile(conf, filePath, "newHadoop.txt");
}
}
```

b. 文件处于被占用状态

```
    public static void main(String[] args) throws IOException, InterruptedException, URISyntaxException {
        Scanner cin = new Scanner(System.in);
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://localhost:9000");
        conf.set("dfs.client.block.write.replace-datanode-on-failure.enable","true");
        conf.set("dfs.client.block.write.replace-datanode-on-failure.policy", "NEVER");
        conf.set("dfs.replication","1");
        String filePath = "/user/hadoop/hadoop.txt";
        String dirPath = "/user/hadoop/data.txt";
        exp10.moveFile(conf, filePath, dirPath);
    }
}
```

c. 文件移动成功

```
    public static void main(String[] args) throws IOException, InterruptedException, URISyntaxException {
        Scanner cin = new Scanner(System.in);
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://localhost:9000");
        conf.set("dfs.client.block.write.replace-datanode-on-failure.enable","true");
        conf.set("dfs.client.block.write.replace-datanode-on-failure.policy", "NEVER");
        conf.set("dfs.replication","1");
        String filePath = "/user/hadoop/hadoop.txt";;
        exp10.moveFile(conf, filePath, "newHadoop.txt");
    }
}
```

实现结果：

a. 文件移动失败

```
Problems  Javadoc  Declaration  Console
<terminated> exp10 [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 12, 2023, 2:37:40 AM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
文件移动失败。
```

b. 文件处于被占用状态

```
<terminated> exp10 [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 12, 2023, 9:32:14 AM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
文件被占用。
```

c. 文件移动成功

```
Problems  Javadoc  Declaration  Console
<terminated> exp10 [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 12, 2023, 9:35:27 AM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
文件移动成功。
```

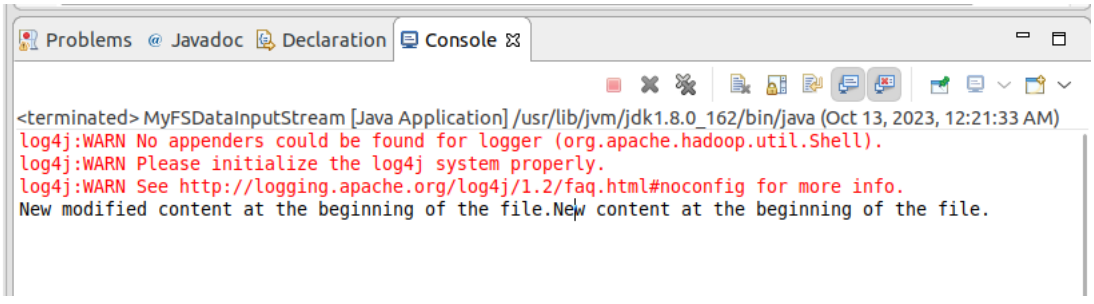
2. 编程实现一个类”MyFSDataInputStream”,该类继承”org.apache.hadoop.fs.FSDataInputStream”，要求：实现按行读取 HDFS 中指定文件的方法“readLine()”，如果读到文件末尾，则返回空，否则返回文件一行的文本。

（方案一）

源代码：

```
import java.io.BufferedReader;

        public class MyFSDataInputStream {
                public static void main(String[] args) {
                        try {
                                Configuration conf = new Configuration();
                                conf.set("fs.defaultFS","hdfs://localhost:9000");
                                conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
                                FileSystem fs = FileSystem.get(conf);
                                Path file = new Path("/user/hadoop/file1.txt");
                                FSDataInputStream getIt = fs.open(file);
                                BufferedReader d = new BufferedReader(new InputStreamReader(getIt));
                                String content = d.readLine(); //读取文件一行
                                if(content != null) {
                                    System.out.println(content);
                                }
                                else {
                                    System.out.println("this is the end of the file");
                                }
                                d.close(); //关闭文件
                                fs.close(); //关闭hdfs
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        }
```

实现结果：



（方案二）

源代码：

```java
    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;
        }
    }
```

MergeFile.java | exp10.java | MyFSDataInputStream.java

```java
     * 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.defaultFS","hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/data.txt";
        try {
            MyFSDataInputStream.cat(conf, remoteFilePath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

实现结果：

Problems | Javadoc | Declaration | Console

```
<terminated> MyFSDataInputStream [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 12, 2023, 5:26:19 AM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
this is data.txt
```

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。

源代码：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FsUrlStreamHandlerFactory;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;

public class ReadFile {

    static {
        URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());
    }

    public static void main(String[] args) throws Exception {
        String hdfsFilePath = "hdfs://localhost:9000/user/hadoop/file1.txt";
        BufferedReader reader = null;
        try {
            URL url = new URL(hdfsFilePath);
            reader = new BufferedReader(new InputStreamReader(url.openStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } finally {
            if (reader != null) {
                reader.close();
            }
        }
    }
}
```

实现结果：

```
<terminated> ReadFile [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (Oct 13, 2023, 12:32:33 AM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.fs.FileSystem).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
New modified content at the beginning of the file.New content at the beginning of the file.
```

五、 实验结果

1. 使用 Java 编程语言和 Shell 命令实现 HDFS 操作

（1） 上传文件到 HDFS：通过编程实现了将本地文本文件上传到 HDFS 的功能。用户可以选择是追加到原有文件末尾还是覆盖原有文件；

（2） 下载文件：编程实现了从 HDFS 中下载指定文件到本地，并在本地文件名与要下载的文件名相同的情况下自动重命名；

（3） 输出 HDFS 文件内容：实现了将 HDFS 中指定文件的内容输出到终端；

（4） 显示 HDFS 文件信息：编程实现了显示 HDFS 中指定文件的读写权限、大小、创建时间、路径等信息；

（5） 显示 HDFS 目录信息：实现了给定 HDFS 中某一个目录，输出该目录下所有文件的读写权限、大小、创建时间、路径等信息，包括递归输出子目录下的文件信息；

（6） 创建和删除 HDFS 文件：编程实现了提供一个 HDFS 内的文件路径，对该文件进行创建和删除操作。如果文件所在目录不存在，程序会自动创建相应目录；

（7） 创建和删除 HDFS 目录：实现了提供一个 HDFS 目录的路径，对该目录进行创建和删除操作。创建目录时，如果目录所在目录不存在，程序会自动创建相应目录。删除目录时，用户可以指定当该目录不为空时是否还删除该目录；

（8） 追加 HDFS 文件内容：编程实现了向 HDFS 中指定的文件追加内容，允许用户选择是追加到文件的开头还是结尾；

（9） 删除 HDFS 文件和目录：实现了删除 HDFS 中指定的文件或目录。对于目录，用户可以指定是否删除包含文件的目录；

（10）移动 HDFS 文件：编程实现了在 HDFS 中将文件从源路径移动到目标路径。

2. 实现自定义 HDFS 文件读取类

实现了一个自定义 HDFS 文件读取类"MyFSDataInputStream"，该类继承自"org.apache.hadoop.fs.FSDataInputStream"，并实现了按行读取 HDFS 中指定文件的方法"readLine()"。这允许用户以更高级的方式按行读取 HDFS 文件的内容。

3. 使用 Java 和 URL 读取 HDFS 文件

编写 Java 程序，使用"java.net.URL"和"org.apache.hadoop.fs.FsURLStreamHandlerFactory"，实现了输出 HDFS 中指定文件的文本到终端的功能。这为从 HDFS 中读取文件并在终端上显示文件内容提供了一种方法。

总的而言，所有实验任务均完成并达到预期。

本次实验使我们能够深入了解 HDFS 的操作和 Hadoop 的相关工具，包括 Shell 命令和自定义 Java 编程。这些技能对于大数据处理和分布式文件系统操作非常重要，有助于我们在大数据领域获得更多经验。

六、 问题和解决

问题 1：

通过 Path 类标识 HDFS 中的文件和目录位置时，可能出现路径错误或无法找到文件的问题。

解决方案：

确保使用正确的 HDFS 路径格式，如 hdfs://<namenode>:<port>/path/to/file，并检查文件或目录是否存在。在调用 FileSystem 类的 copyFromLocalFile 方法之前，可以使用 Path 构造器创建正确的路径对象。

问题 2：

通过 FileSystem 类的 copyFromLocalFile 方法将文件从本地文件系统复制到 Hadoop 文件系统时，可能会出现复制失败或权限问题。

解决方案：

确保指定的本地文件存在，HDFS 路径有效，且有足够的权限执行复制操作。检查 Hadoop 集群的连接是否正常，确保网络通畅。还要注意端口和配置是否正确。

问题 3：

使用 FileSystem 的 copyToLocalFile 方法从 HDFS 中的 remotePath 下载文件到新的本地路径 localPath 时，可能出现复制失败或文件不存在的问题。

解决方案：

确保 HDFS 中的 remotePath 存在并且包含正确的文件名，localPath 指定了有效的本地目录。还要注意权限问题，确保您有权限读取 HDFS 中的文件。

问题 4：

使用 FileSystem 对象的 open 方法打开文件后，可能遇到无法读取文件内容的问题。

解决方案：

确保文件存在并且指定的路径正确。如果文件不存在，将无法打开它。如果路径正确，但读取失败，可以检查 HDFS 权限是否足够。另外，确保 Hadoop 集群运行正常。

问题 5：

在创建 InputStreamReader 和 BufferedReader 时可能会出现编码问题或流处理错误。

解决方案：

确保使用正确的字符编码，通常是 UTF-8。您还应确保正确处理异常，如在创建 InputStreamReader 时捕获 IOException。使用 BufferedReader 允许逐行读取文本，这是一种更高效的方式。

问题 6：

Hadoop 环境配置不正确，导致 HDFS 操作失败。

解决方案：

确保 Hadoop 的配置文件（如 core-site.xml 和 hdfs-site.xml）正确设置，包括 HDFS 的 URI 和相关属性。检查 Java 环境变量和 Hadoop 安装路径是否正确。

问题 7：

在使用 Java URL 处理 HDFS 文件时出现问题。

解决方案：

确保使用正确的 URL 格式来引用 HDFS 文件。检查 URL 处理程序是否正确设置。查看 Java 文档和 Hadoop 文档以获取更多信息，如果遇到问题仍无法解决，可以寻求 Hadoop 社区或论坛的帮助。