**中 国 地 质 大 学**

**本科生课程论文**



课程名称 大数据技术基础

教师姓名 卢超

学生姓名 常文瀚

学生学号 20181001095

学生班级 191181

所在学院 计算机学院

完成日期 2021年6月20日

目录

**第一章 熟悉常用的HDFS操作1**

1.1实验目的1

1.2实验平台1

1.3实验内容和要求1

1.4实验过程2

1.5实验中的目的和解决方法7

**第二章 熟悉常用的HBase操作9**

2.1实验目的9

2.2实验平台9

2.3实验内容和要求10

2.4实验过程11

2.5实验中的目的和解决方法15

**第三章** **MapReduce编程初级实践15**

3.1实验目的15

3.2实验平台15

3.3实验内容和要求15

3.4实验过程19

3.5实验中的目的和解决方法25

**第四章 总结与体会29**

**第五章 参考与引用30**

**附录一.** **MapReduce编程初级实践实验代码31**

第一章 熟悉常用的HDFS操作

# 1.1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；

# 1.2实验平台

操作系统：Linux

操作平台：Docker

Hadoop版本：2.7.1或以上版本

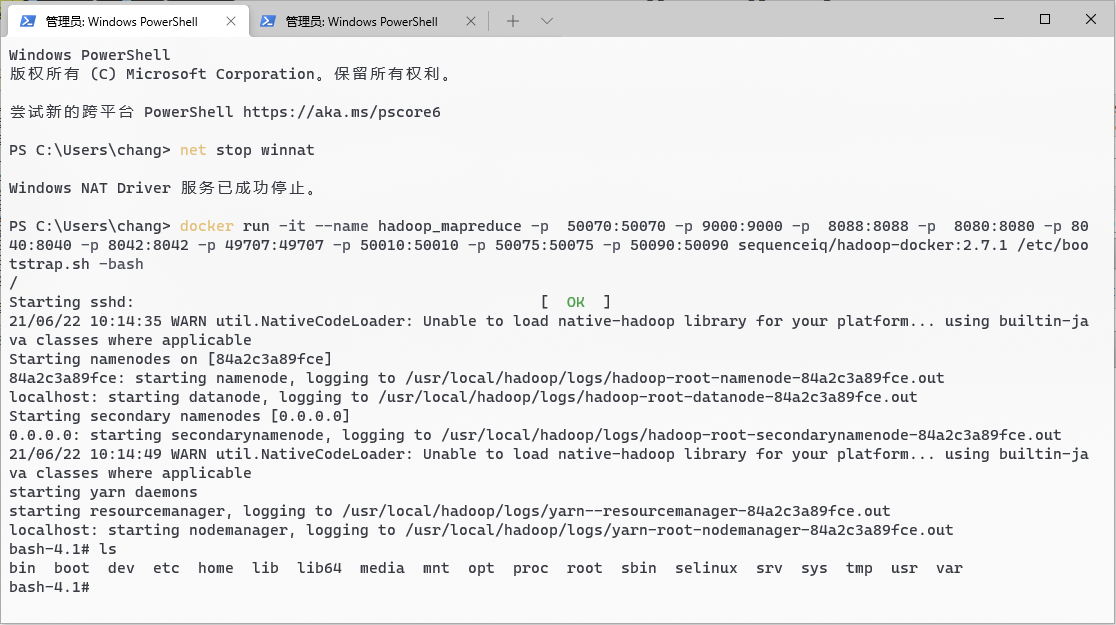
JDK版本：1.8或以上版本

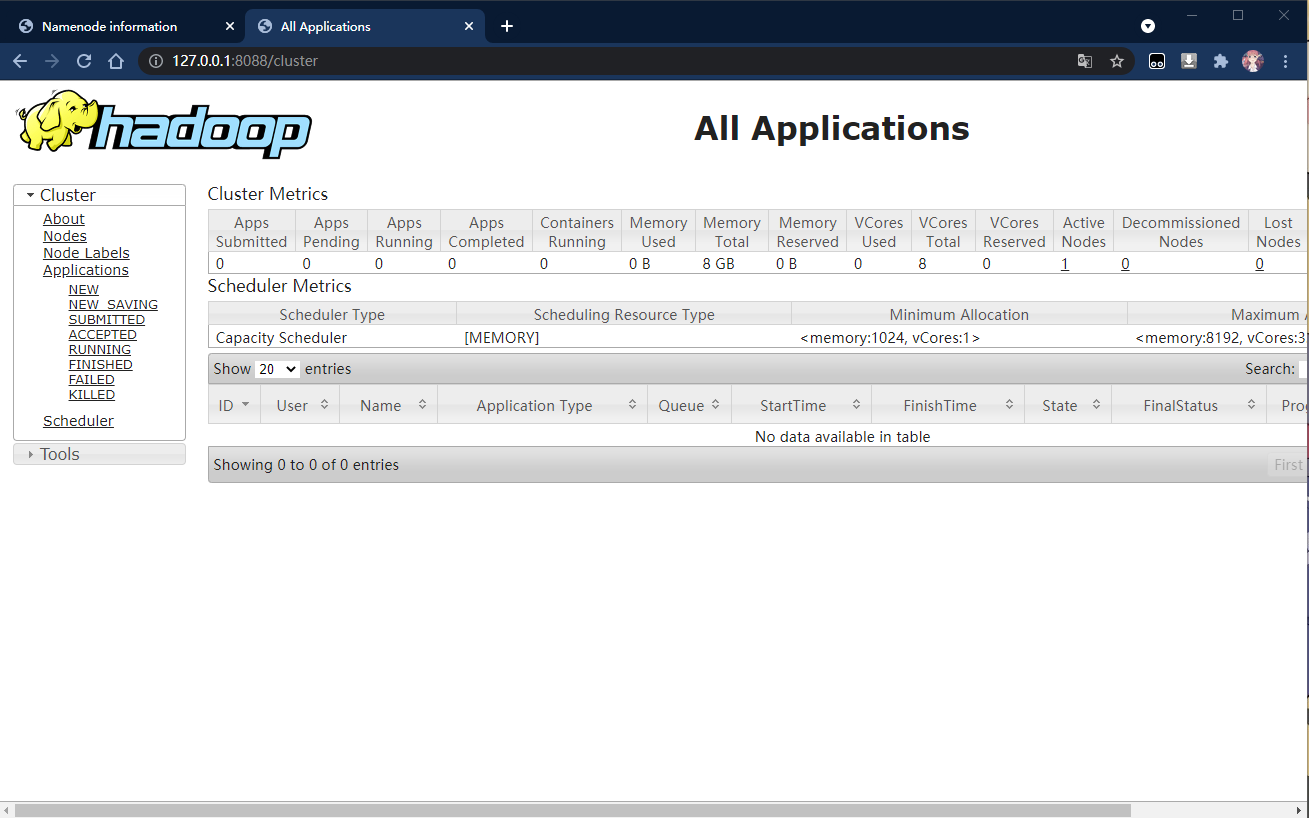
# 1.3实验内容和要求

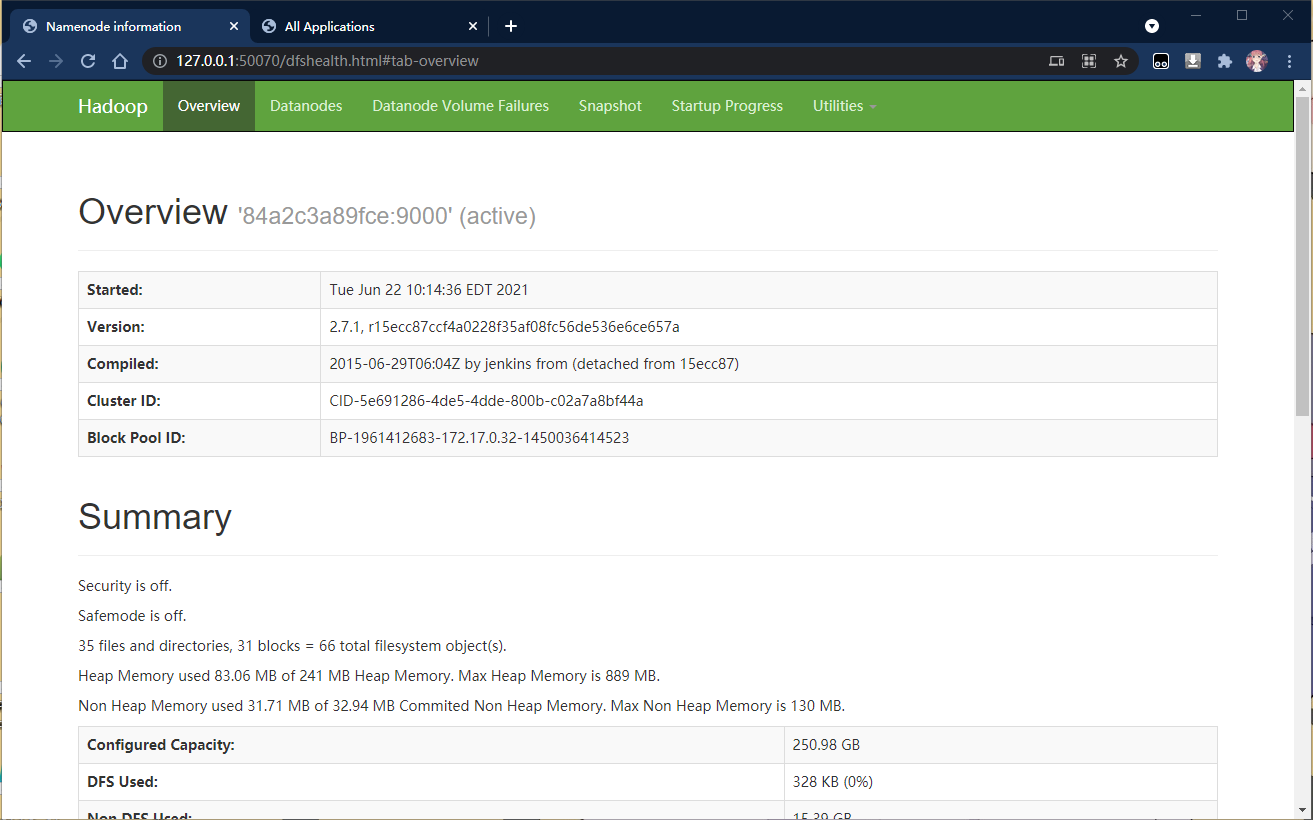
1. 编程实现以下指定功能，并利用Hadoop提供的Shell命令完成相同任务：
2. 向HDFS中上传任意文本文件，如果指定的文件在HDFS中已经存在，由用户指定是追加到原有文件末尾还是覆盖原有的文件；
3. 从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名；
4. 将HDFS中指定文件的内容输出到终端中；
5. 显示HDFS中指定的文件的读写权限、大小、创建时间、路径等信息；
6. 给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，如果该文件是目录，则递归输出该目录下所有文件相关信息；
7. 提供一个HDFS内的文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

# 1.4实验过程

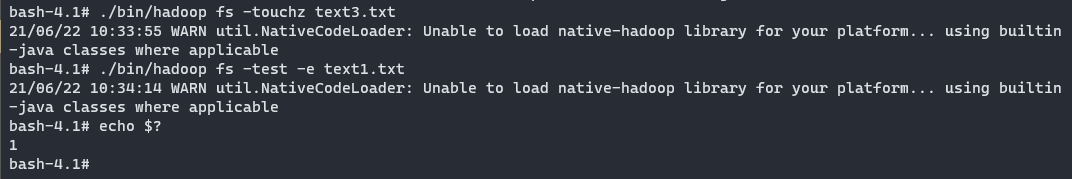
（1）安装Docker，拉取在线镜像，并且运行。

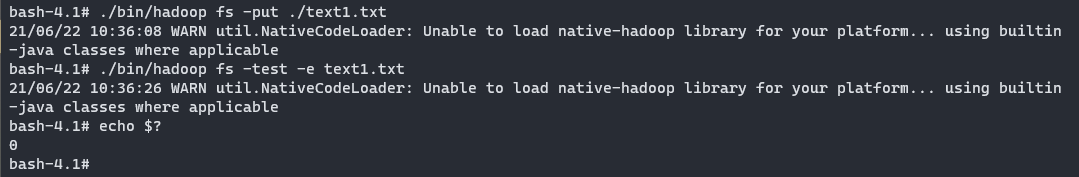


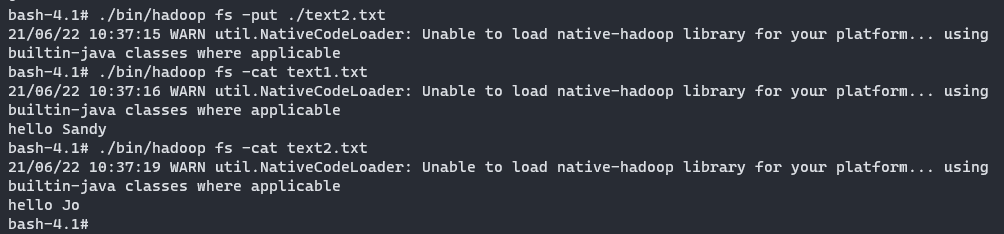


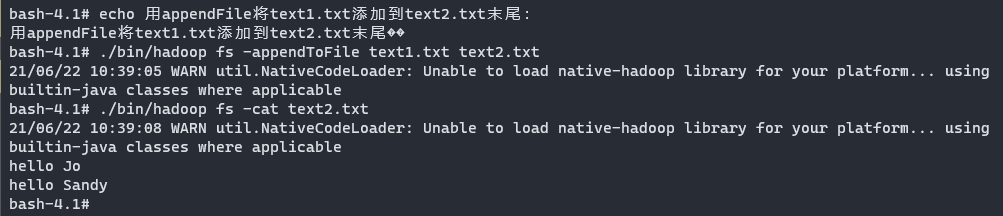


（2）查看文件是否存在于HDFS，并编辑新的文本文件，将其上传，读取文件后将内容添加在末尾，并显示文本内容。

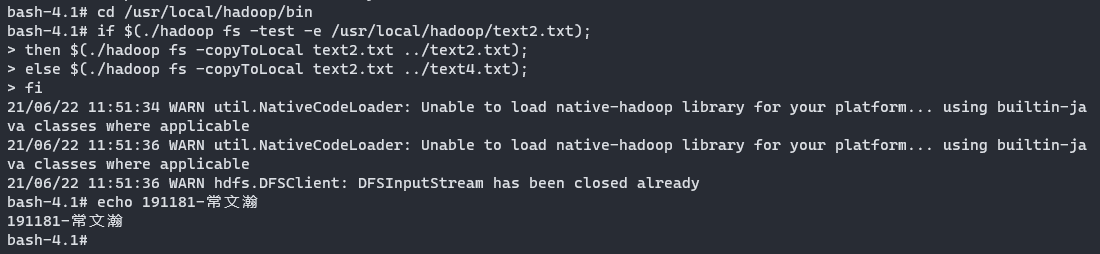




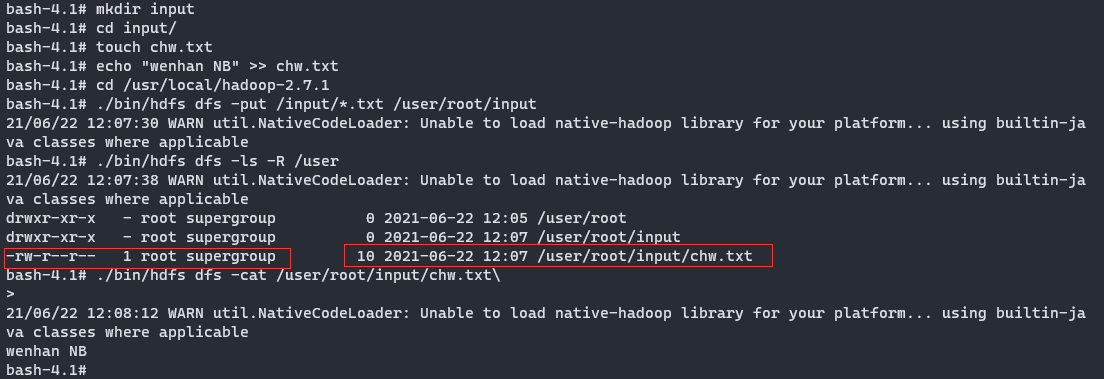




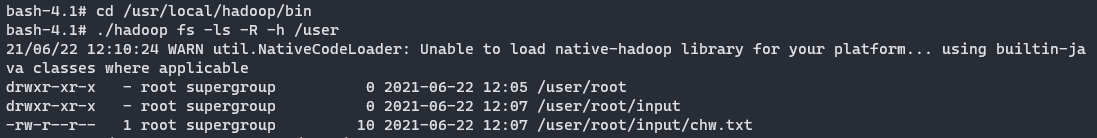
（3）从HDFS中下载指定文件，如果本地文件与要下载的文件名称相同，则自动对下载的文件重命名。

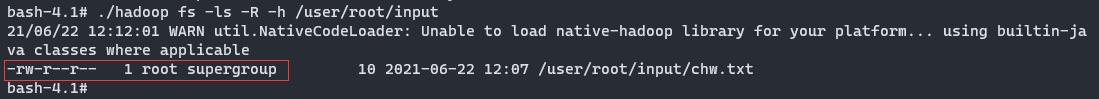


（4）显示HDFS中指定的文件的读写权限、大小、创建时间、路径等信息。

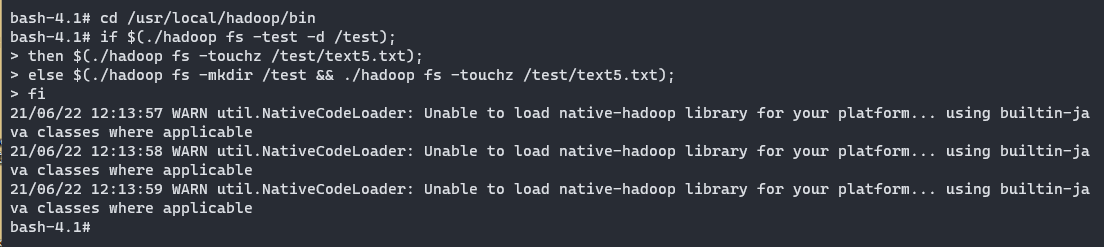


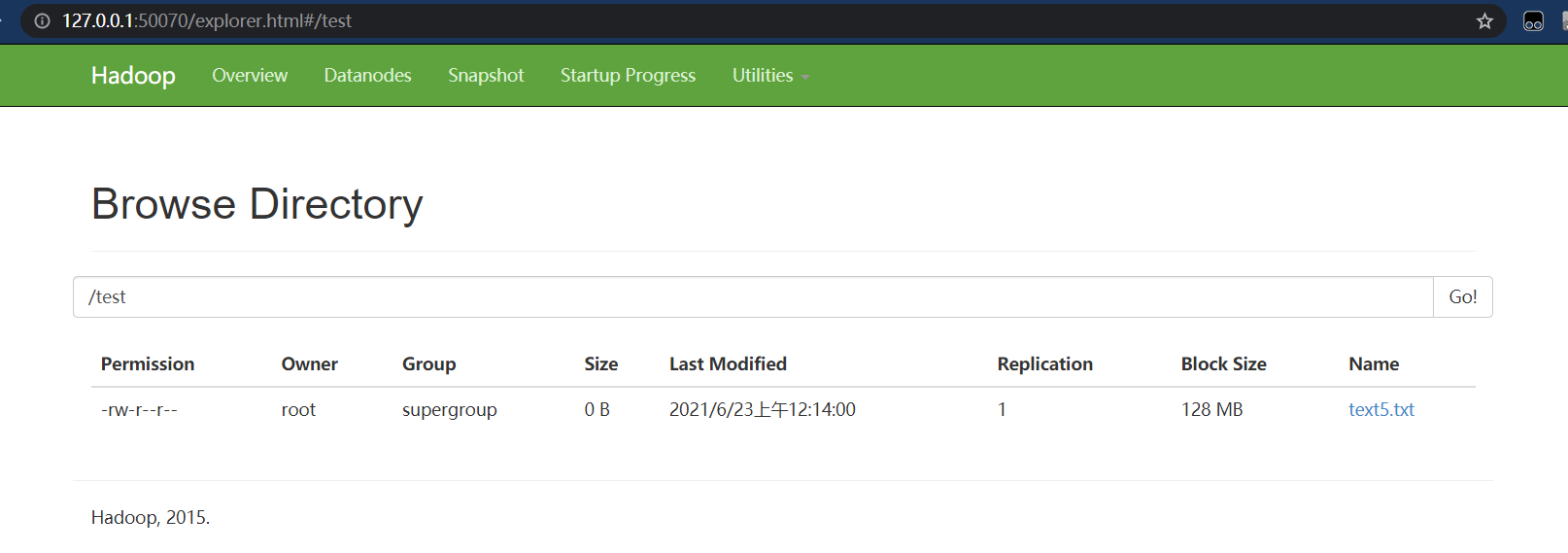
（5）给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，如果该文件是目录，则递归输出该目录下所有文件相关信息。

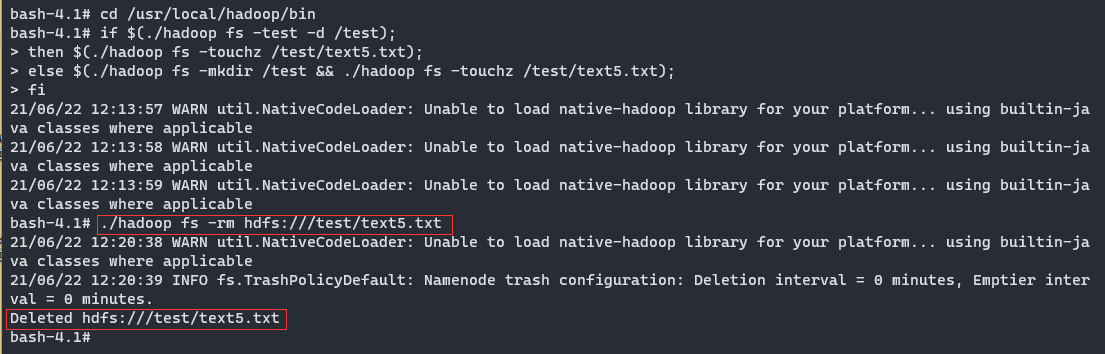


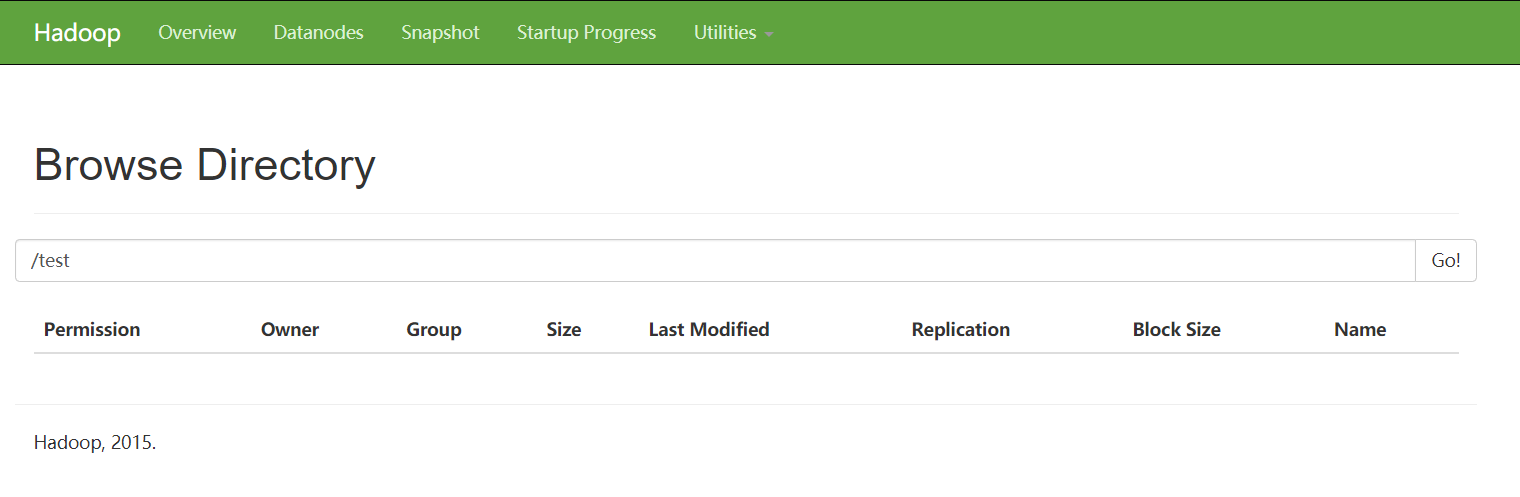


（6）提供一个HDFS内的文件的路径，对该文件进行创建和删除操作。如果文件所在目录不存在，则自动创建目录。

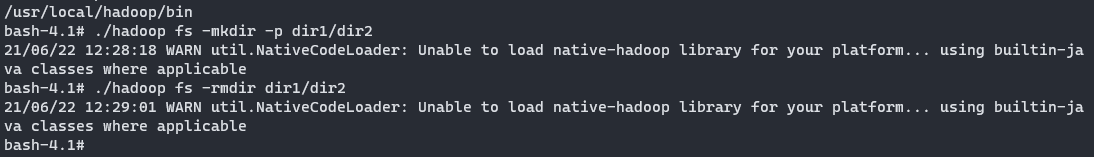


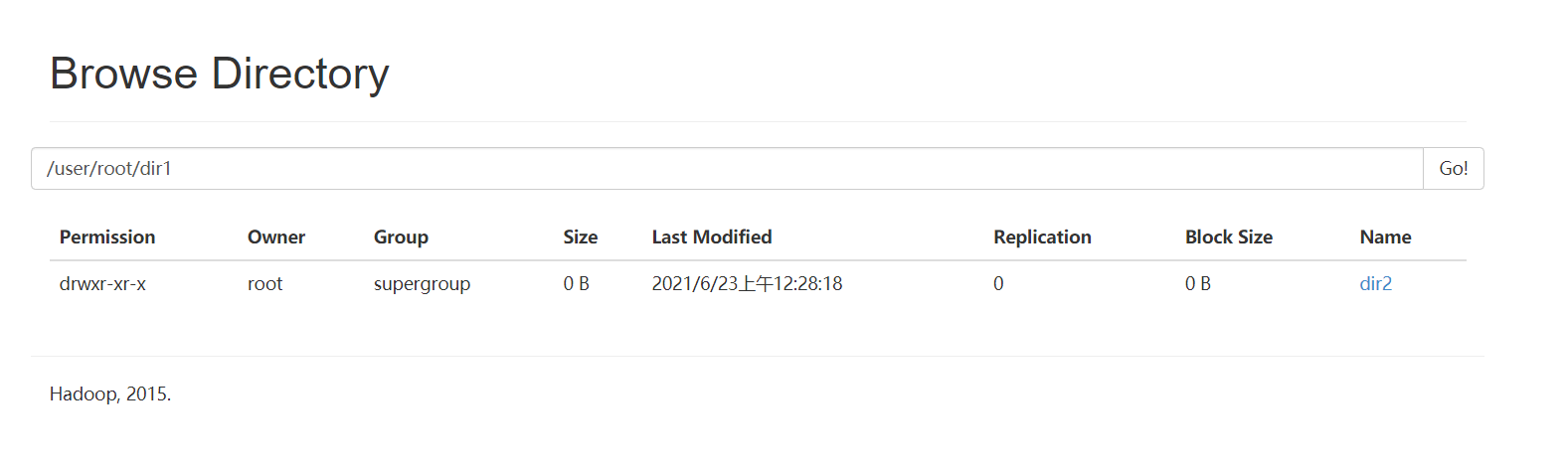


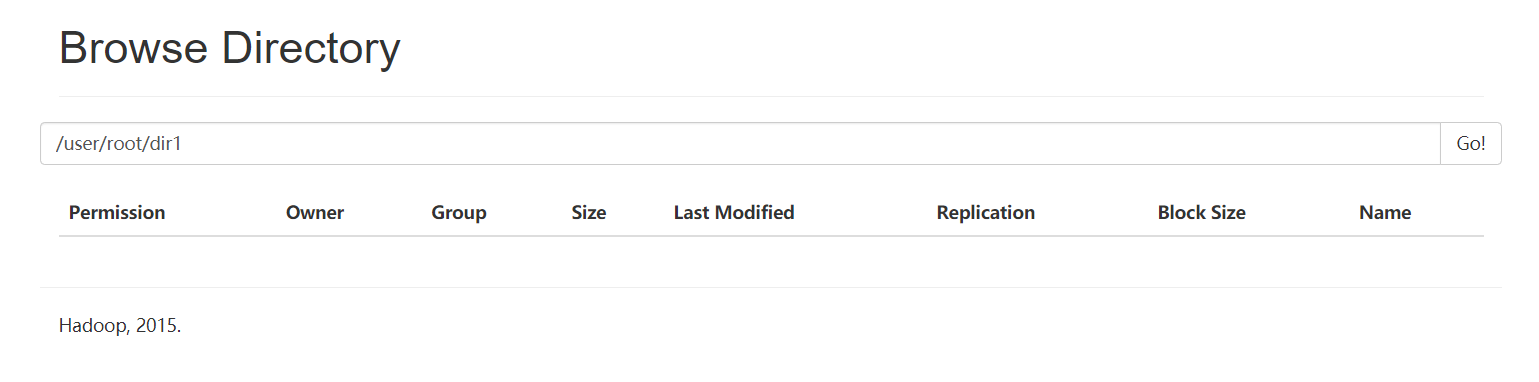




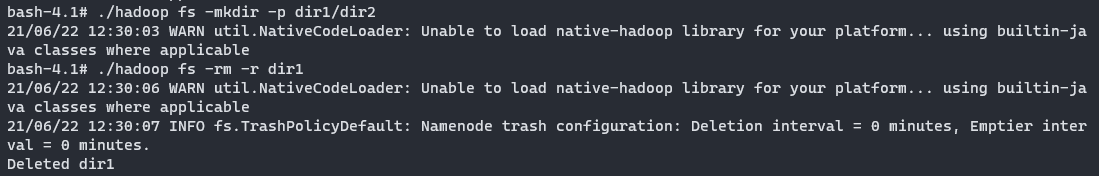
（7）提供一个HDFS的目录的路径，对该目录进行创建和删除操作，创建目录时，如果目录文件所在目录不存在则自动创建相应目录，删除目录时，由用户指定当该目录不为空时是否还删除该目录。

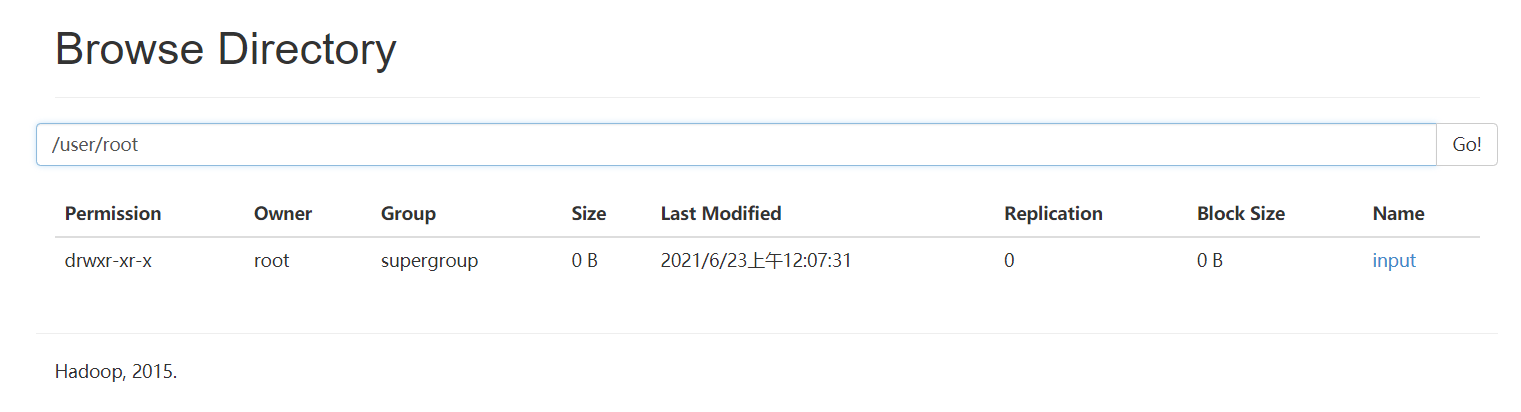






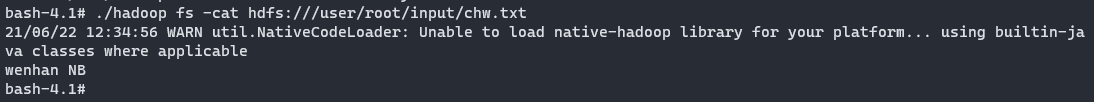
#rmdir只能删除空目录，不能删除非空目录。

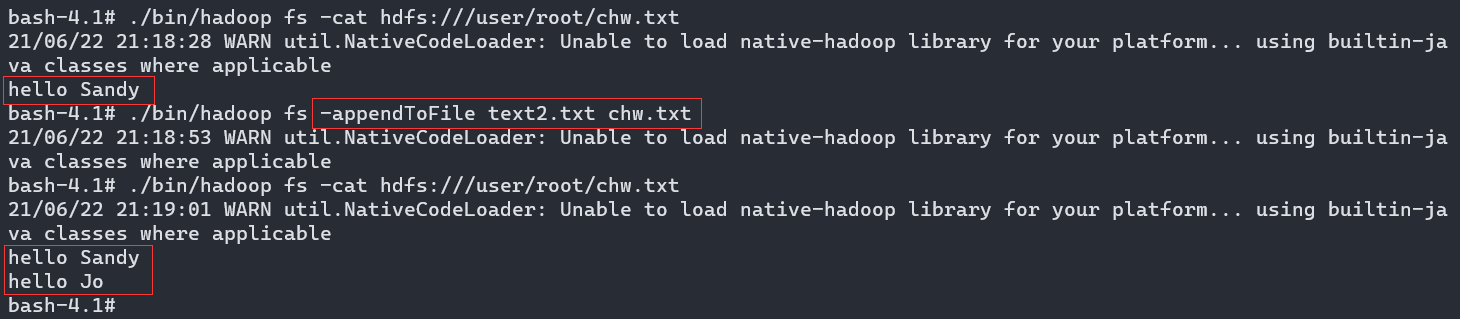


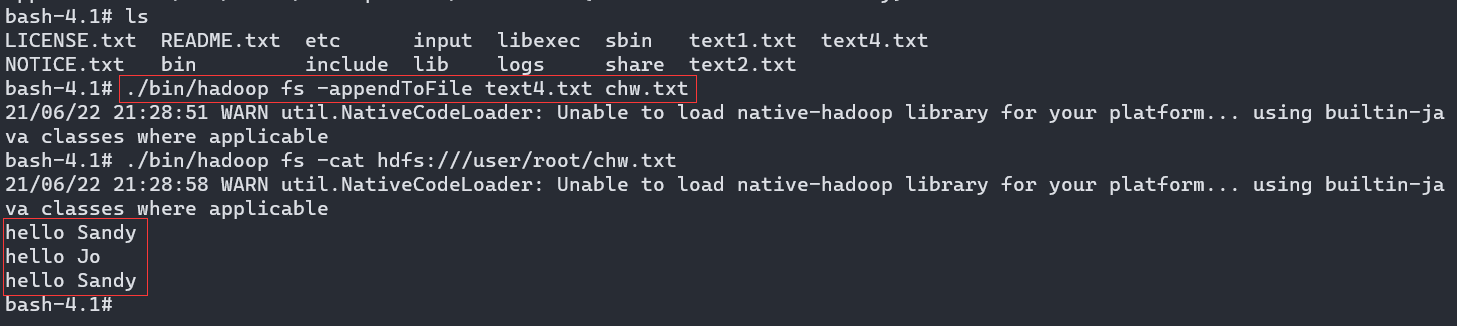


dir1/dir2全部被删除

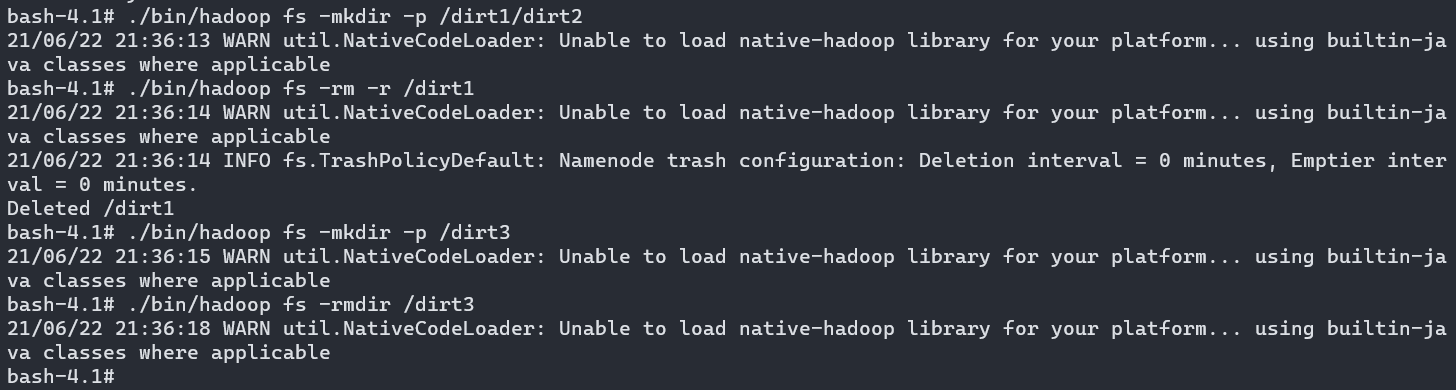
（8）向HDFS中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾。



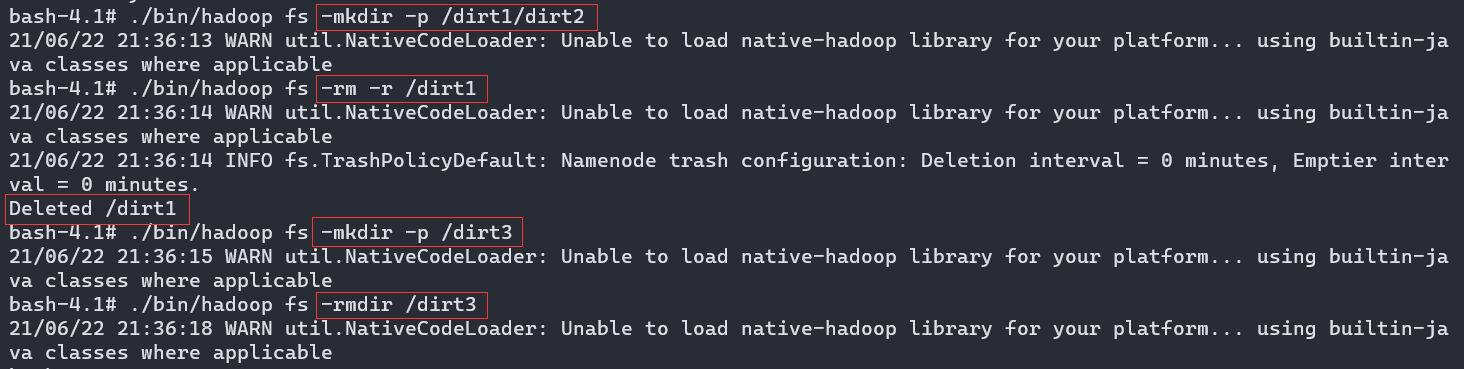




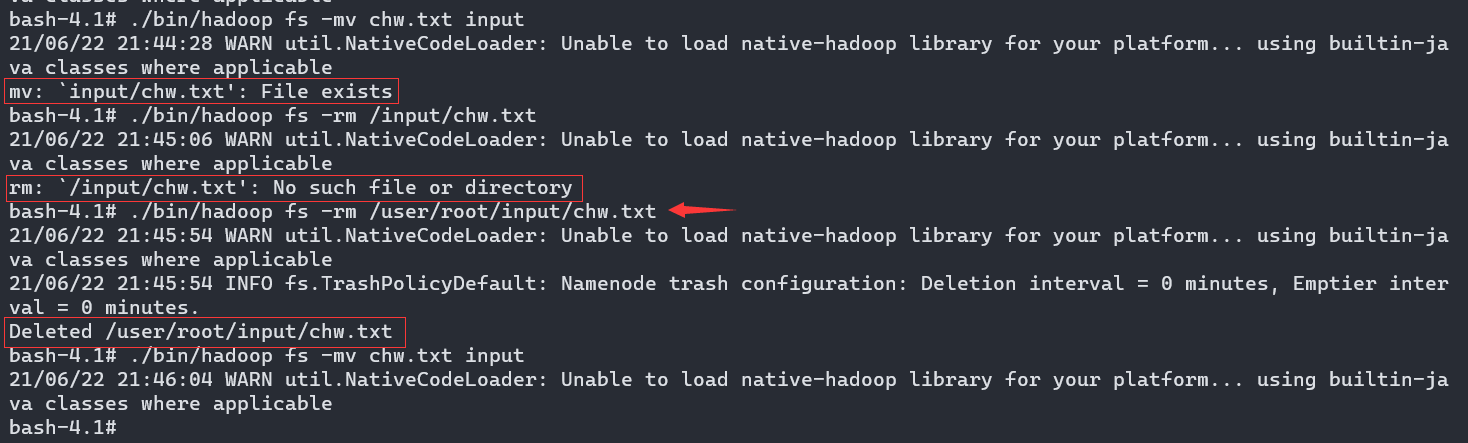
（9）删除HDFS中指定的文件



（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；

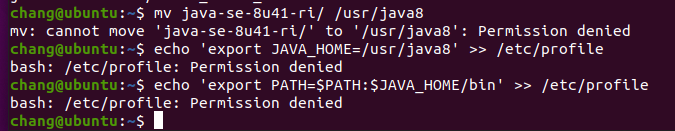


（11）在HDFS中将文件从源路径移动到目的路径。



# 1.5实验中的问题与解决方法

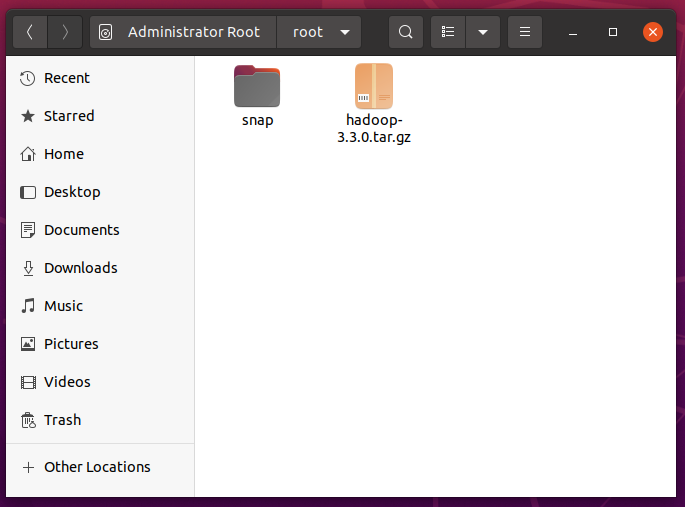
（1）配置java环境时，配置环境路径失败。



原因：修改路径等信息时，需要Ubuntu管理员权限。

解决方法：使用sudo su，输入系统密码，获得管理员权限进行修改。

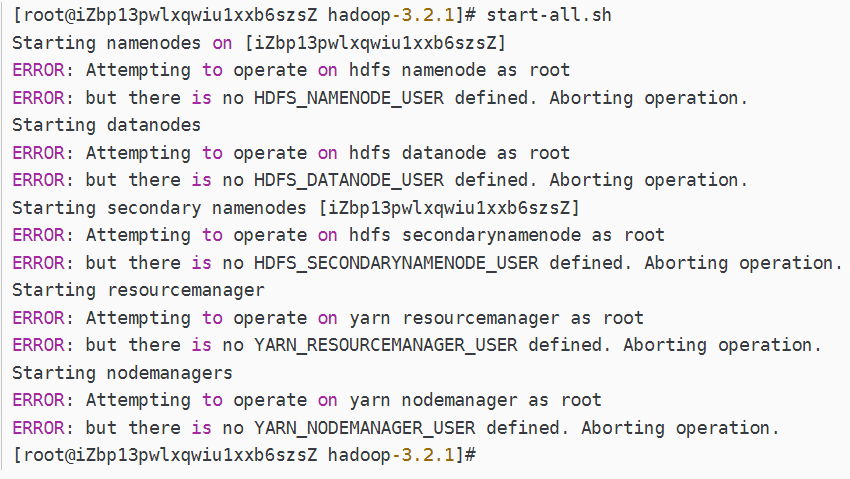
（2）下载Hadoop时，下载速度过慢。



原因：软件源设置到了Ubuntu默认软件源。

解决方法：可以使用国内镜像或者下载到Windows，再上传到Ubuntu上。

（3）执行start-dfs.sh start-yarn.sh两个文件会报错，例如：



原因：Hadoop为不同的用户安装，你为不同的用户启动yarn服务，或者是在 Hadoop配置的hadoop-env.sh中指定了HDFS\_NAMENODE\_USER但是HDFS\_DATANODE\_USER 用户是别的东西。

解决方法：输入下列代码，配置用户为root



（4）安装后8088端口可以访问50070无法访问，防火墙开放后仍然无法访。

原因：Namenode初始化默认端口失效，需要修改配置文件。

解决方法：手动修改配置文件设置默认端口，hdfs-site.xml 添加如下代码。



第二章 熟悉常用的HBase操作

# 2.1实验目的

1. 理解HBase在Hadoop体系结构中的角色；

2. 熟练使用HBase操作常用的Shell命令；

3. 熟悉HBase操作常用的Java API（选做）。

# 2.2实验平台

操作系统：Linux

Hadoop版本：2.7.1或以上版本

HBase版本：1.1.2或以上版本

JDK版本：1.8或以上版本

Java IDE：未安装桌面系统，使用vim编辑代码

# 2.3实验内容和要求

1. 编程实现以下指定功能，并用Hadoop提供的HBase Shell命令完成相同任务：
2. 列出HBase所有的表的相关信息，例如表名；
3. 在终端打印出指定的表的所有记录数据；
4. 向已经创建好的表添加和删除指定的列族或列；
5. 清空指定的表的所有记录数据；
6. 统计表的行数。
7. 现有以下关系型数据库中的表和数据，要求将其转换为适合于HBase存储的表并插入数据：

学生表（Student）

|  |  |  |  |
| --- | --- | --- | --- |
| 学号（S\_No） | 姓名（S\_Name） | 性别（S\_Sex） | 年龄（S\_Age） |
| 2015001 | Zhangsan | male | 23 |
| 2015003 | Mary | female | 22 |
| 2015003 | Lisi | male | 24 |

课程表（Course）

|  |  |  |
| --- | --- | --- |
| 课程号（C\_No） | 课程名（C\_Name） | 学分（C\_Credit） |
| 123001 | Math | 2.0 |
| 123002 | Computer Science | 5.0 |
| 123003 | English | 3.0 |

选课表（SC）

|  |  |  |
| --- | --- | --- |
| 学号（SC\_Sno） | 课程号（SC\_Cno） | 成绩（SC\_Score） |
| 2015001 | 123001 | 86 |
| 2015001 | 123003 | 69 |
| 2015002 | 123002 | 77 |
| 2015002 | 123003 | 99 |
| 2015003 | 123001 | 98 |
| 2015003 | 123002 | 95 |

1. 利用HBase和MapReduce完成如下任务：

假设HBase有2张表，表的逻辑视图及部分数据如下所示：

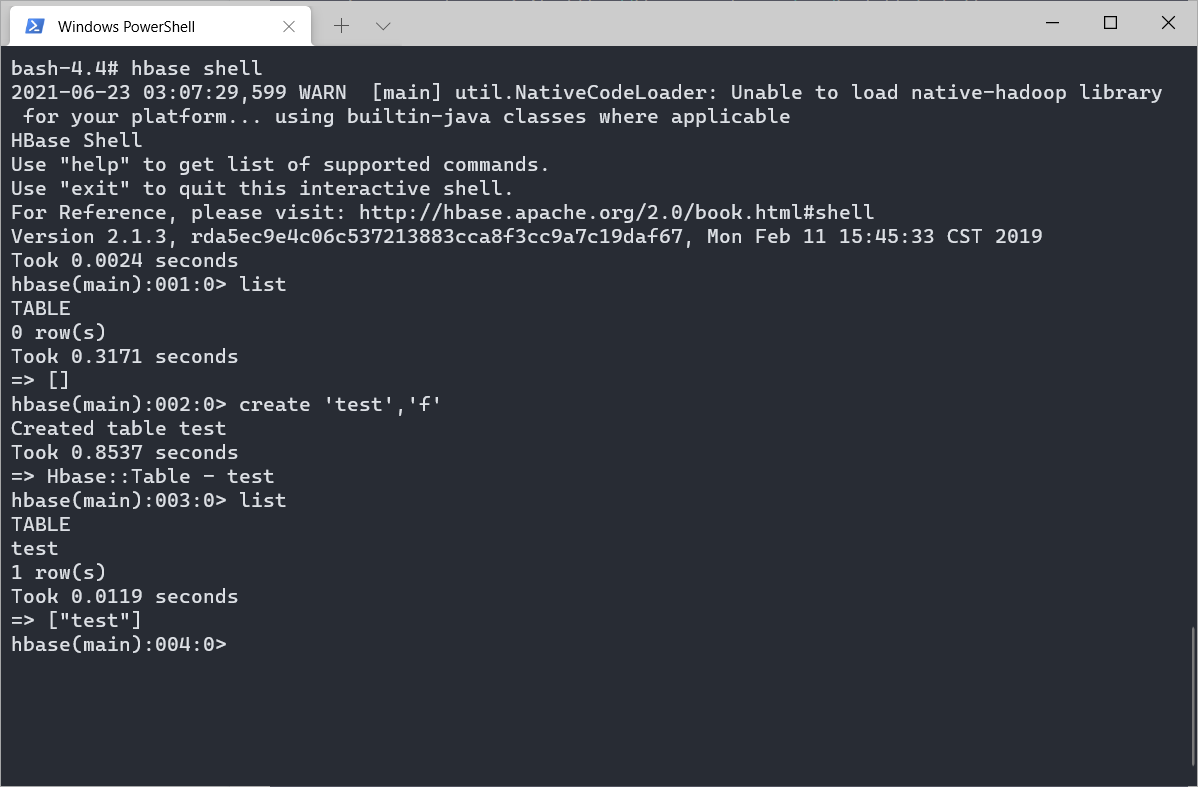
表 逻辑视图及部分数据

|  |  |
| --- | --- |
| 书名（bookName） | 价格（price） |
| Database System Concept | 30$ |
| Thinking in Java | 60$ |
| Data Mining | 25$ |

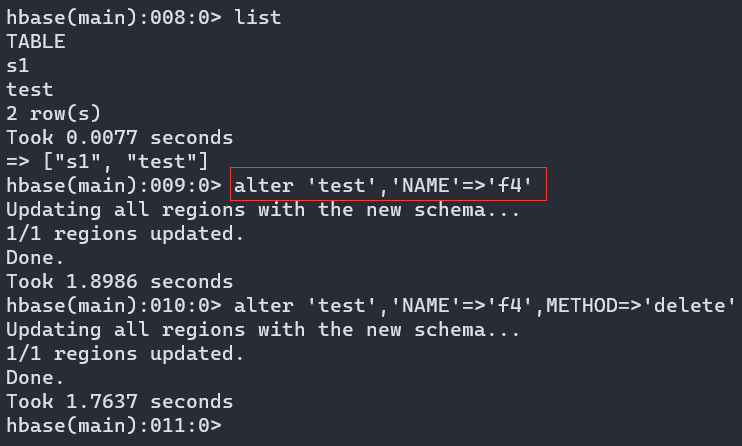
要求：从HBase读出上述两张表的数据，对“price”的排序，并将结果存储到HBase中。

**2.4实验过程**

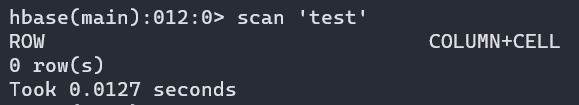
（1）创建一个表，并查看Linux上的表名称

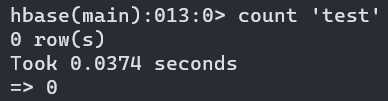


（2）添加信息，删除信息

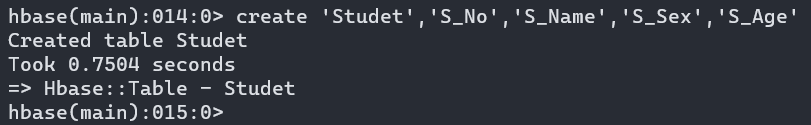


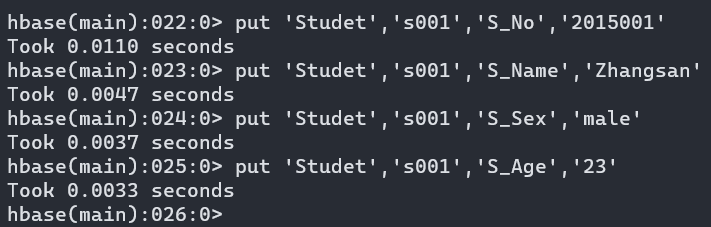
（3）扫描表，统计行数



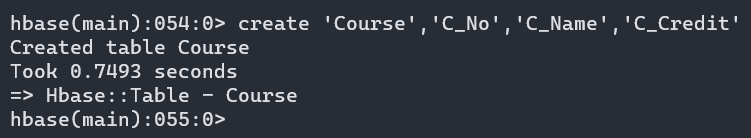


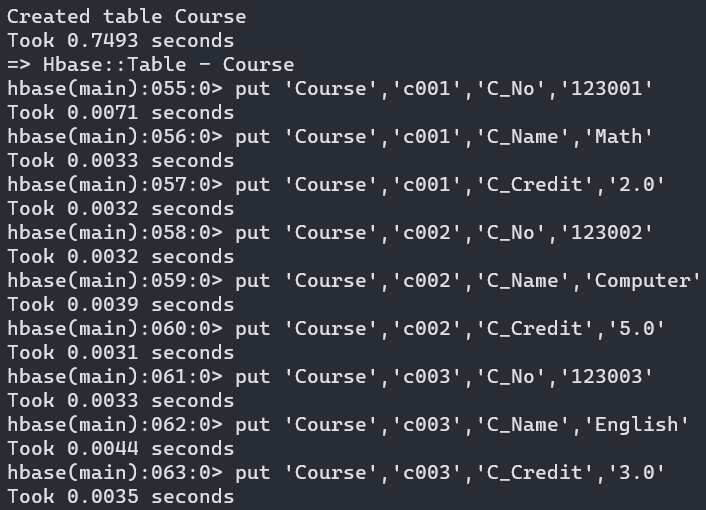
（4）创建Student表，并添加数据



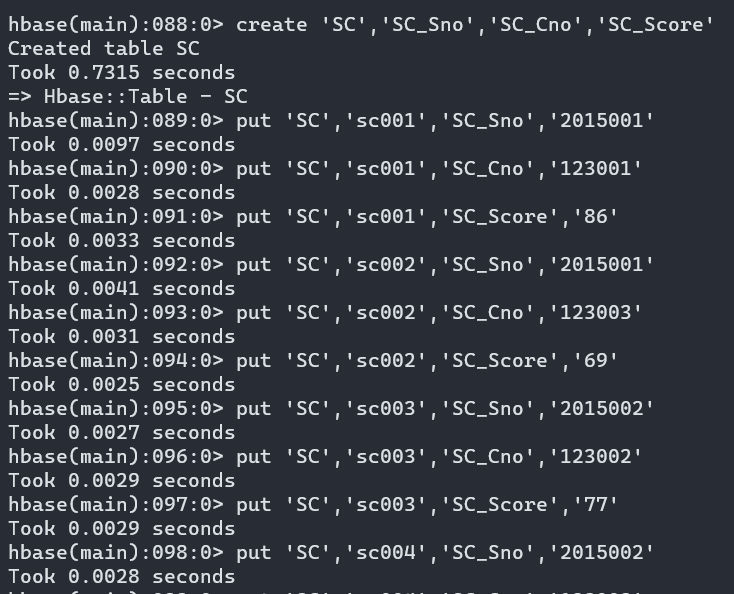


（5）创建Course表，添加课程

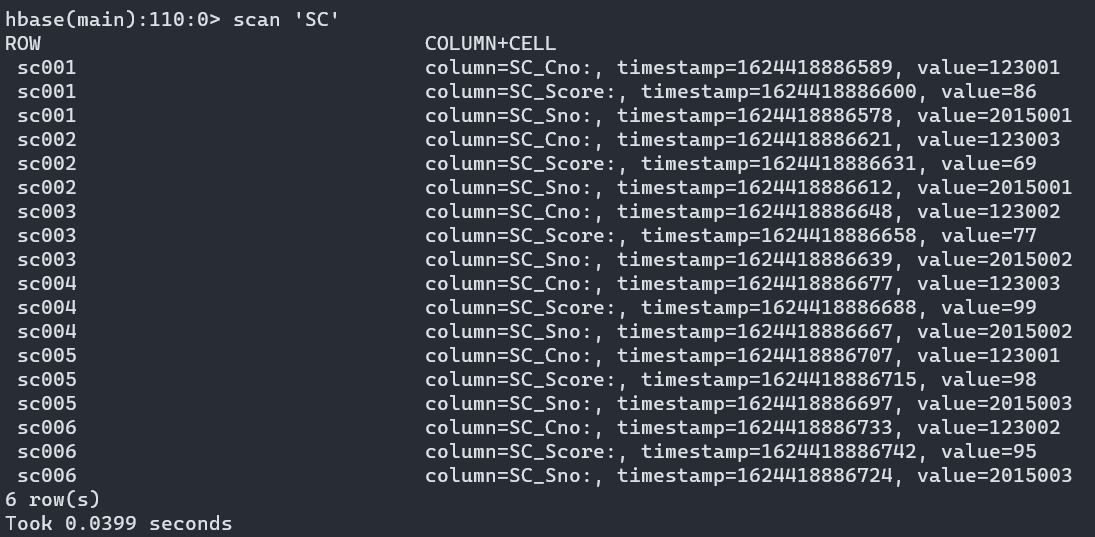




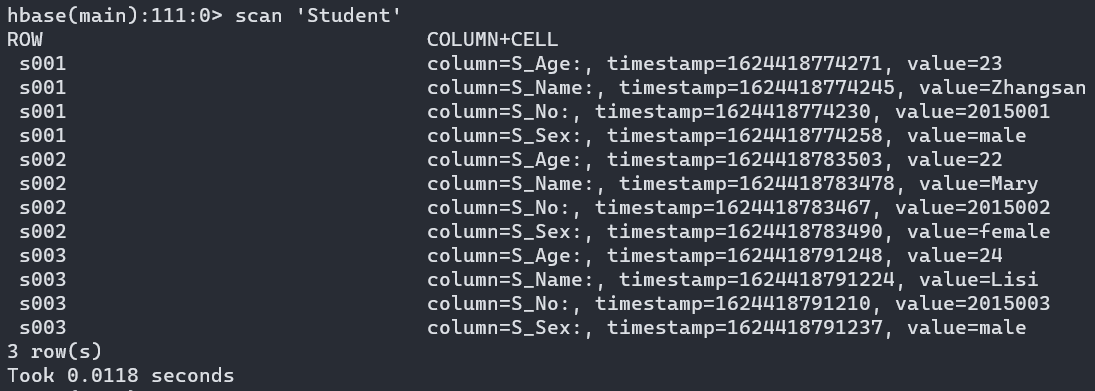
（6）创建SC表，添加信息



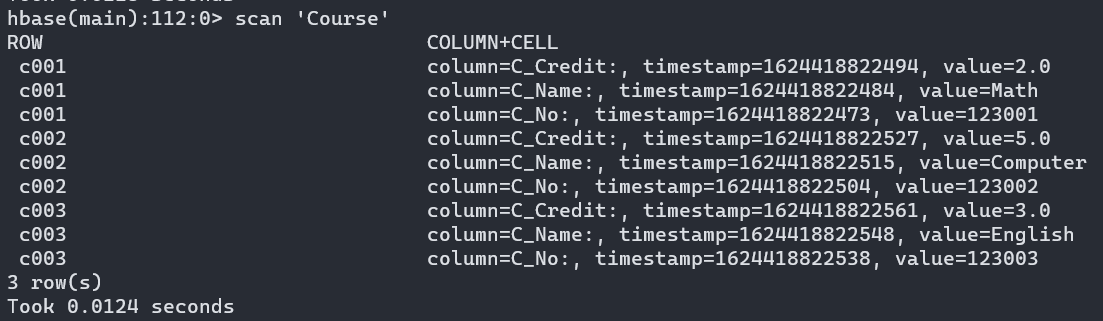
（7）打印SC表



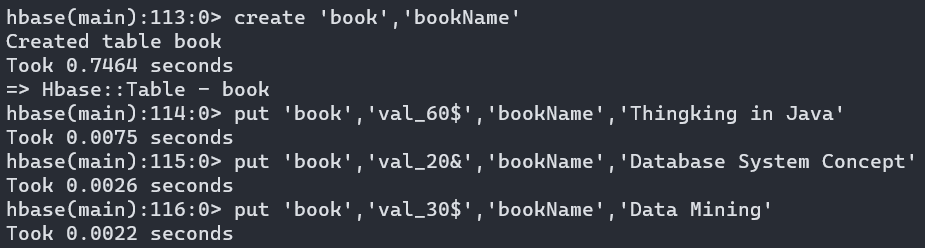
（8）打印Student表

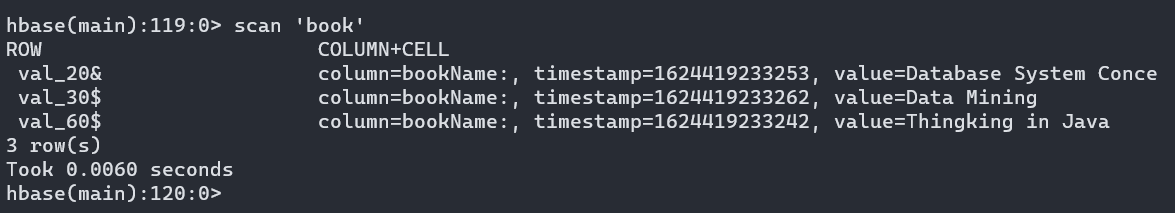


（9）打印Course表



（10）创建书籍表，直接打印，可以根据设置的value自动排序





**2.5****实验中的问题与解决方法**

（1）启动Docker，运行start-hbase失败

原因：在启动Docker时，封装好的Hbase直接运行了起来，所以不需要手动启动。

解决方法：Hbase已运行，可以直接操作。

第三章 MapReduce编程初级实践

**3.1实验目的**

1.通过实验掌握基本的MapReduce编程方法；

2.掌握用MapReduce解决一些常见的数据处理问题，包括数据去重、数据排序和数据挖掘等。

**3.2实验平台**

已经配置完成的Hadoop伪分布式环境。

**3.3实验内容和要求**

## 1.编程实现文件合并和去重操作

对于两个输入文件，即文件A和文件B，请编写MapReduce程序，对两个文件进行合并，并剔除其中重复的内容，得到一个新的输出文件C。下面是输入文件和输出文件的一个样例供参考。

输入文件A的样例如下：

|  |
| --- |
| 20150101 x  20150102 y  20150103 x  20150104 y  20150105 z  20150106 x |

输入文件B的样例如下：

|  |
| --- |
| 20150101 y  20150102 y  20150103 x  20150104 z  20150105 y |

根据输入文件A和B合并得到的输出文件C的样例如下：

|  |
| --- |
| 20150101 x  20150101 y  20150102 y  20150103 x  20150104 y  20150104 z  20150105 y  20150105 z  20150106 x |

## 2. 编写程序实现对输入文件的排序

现在有多个输入文件，每个文件中的每行内容均为一个整数。要求读取所有文件中的整数，进行升序排序后，输出到一个新的文件中，输出的数据格式为每行两个整数，第一个数字为第二个整数的排序位次，第二个整数为原待排列的整数。下面是输入文件和输出文件的一个样例供参考。

输入文件1的样例如下：

|  |
| --- |
| 33  37  12  40 |

输入文件2的样例如下：

|  |
| --- |
| 4  16  39  5 |

输入文件3的样例如下：

|  |
| --- |
| 1  45  25 |

根据输入文件1、2和3得到的输出文件如下：

|  |
| --- |
| 1 1  2 4  3 5  4 12  5 16  6 25  7 33  8 37  9 39  10 40  11 45 |

## 3. 对给定的表格进行信息挖掘

下面给出一个child-parent的表格，要求挖掘其中的父子辈关系，给出祖孙辈关系的表格。

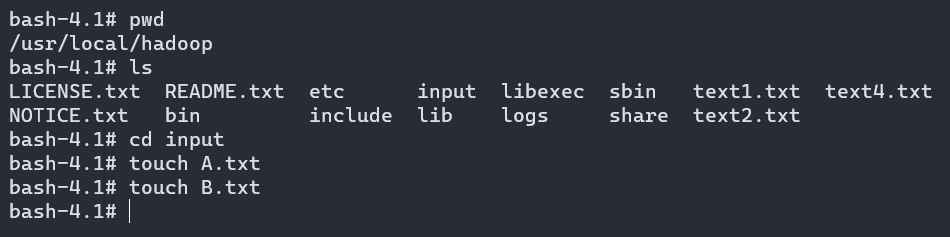
输入文件内容如下：

|  |
| --- |
| child parent  Steven Lucy  Steven Jack  Jone Lucy  Jone Jack  Lucy Mary  Lucy Frank  Jack Alice  Jack Jesse  David Alice  David Jesse  Philip David  Philip Alma  Mark David  Mark Alma |

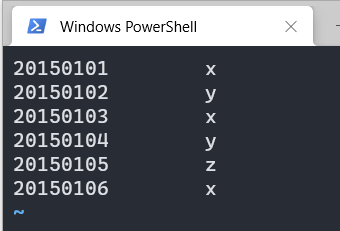
输出文件内容如下：

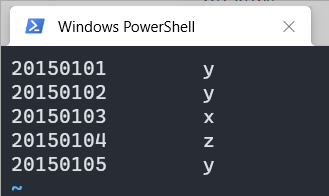
|  |
| --- |
| grandchild grandparent  Steven Alice  Steven Jesse  Jone Alice  Jone Jesse  Steven Mary  Steven Frank  Jone Mary  Jone Frank  Philip Alice  Philip Jesse  Mark Alice  Mark Jesse |

**3.4实验过程**

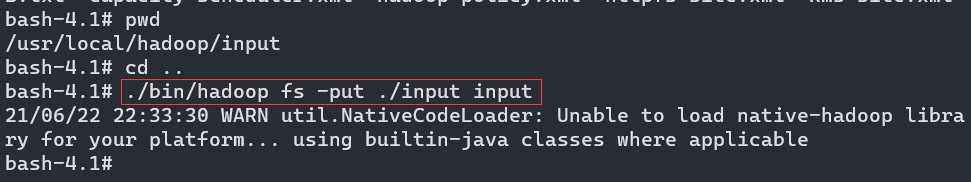


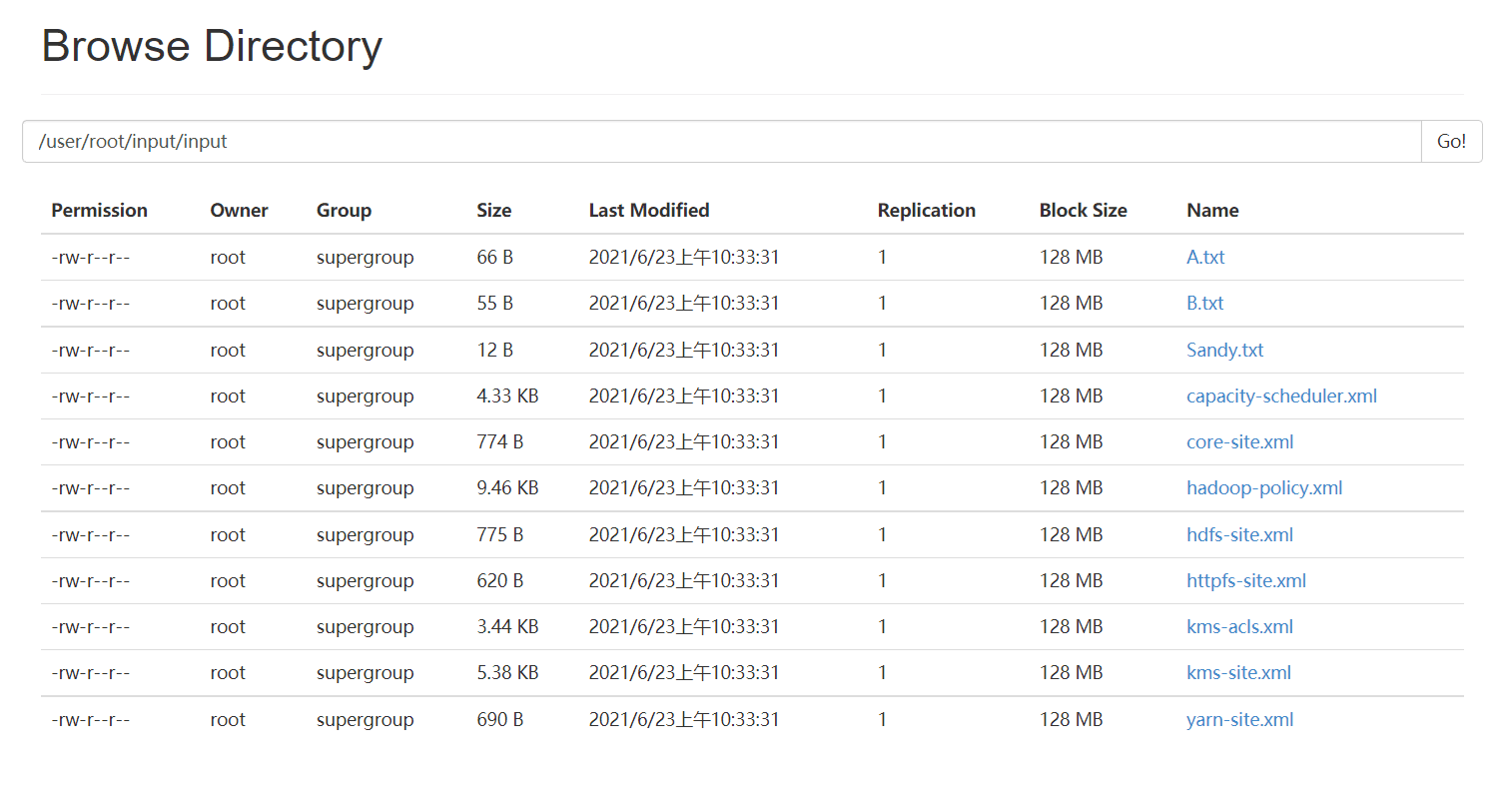
（2）创建文本并输入信息



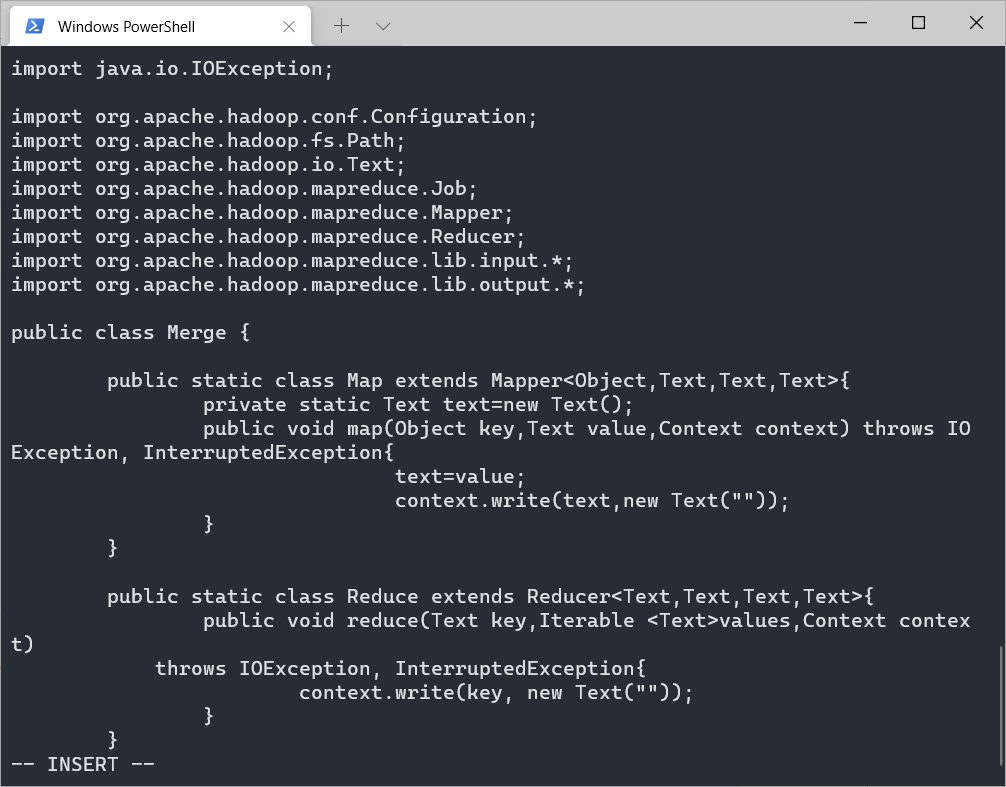


（3）上传文件

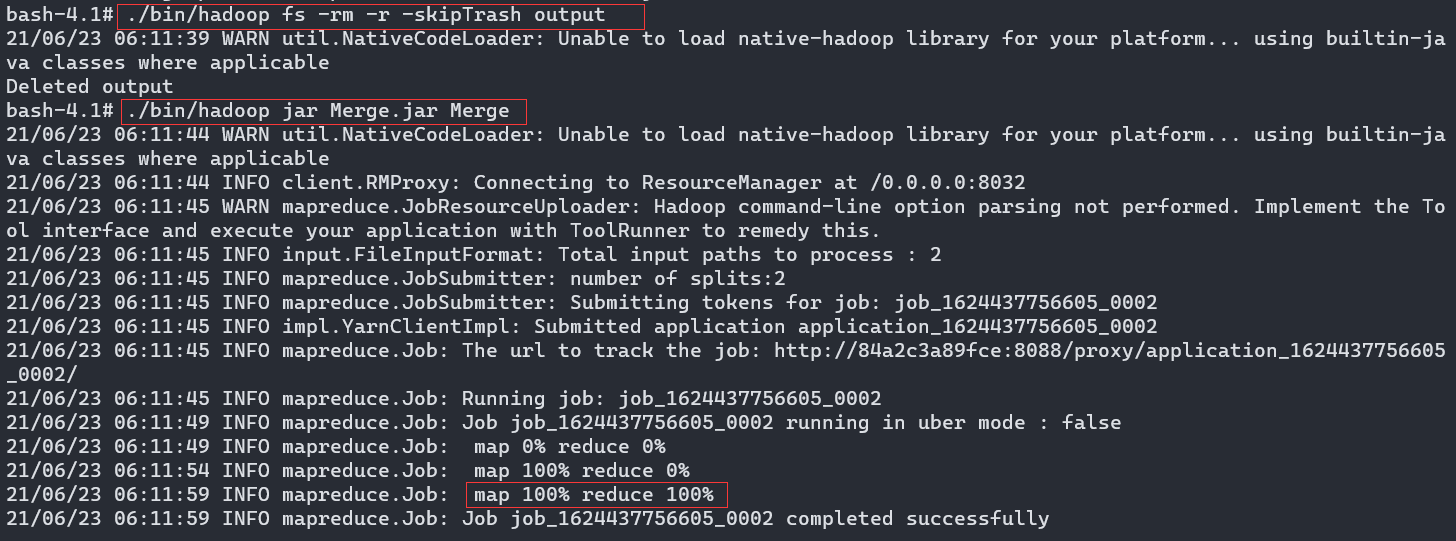




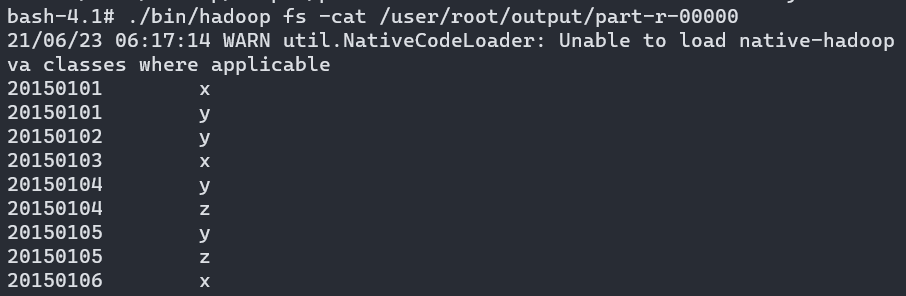
（4）编辑代码



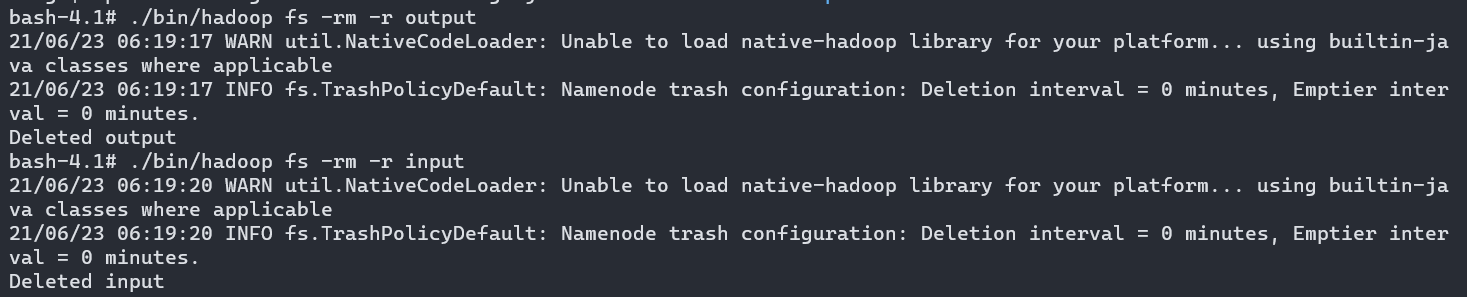
（5）编译java代码，运行jar包



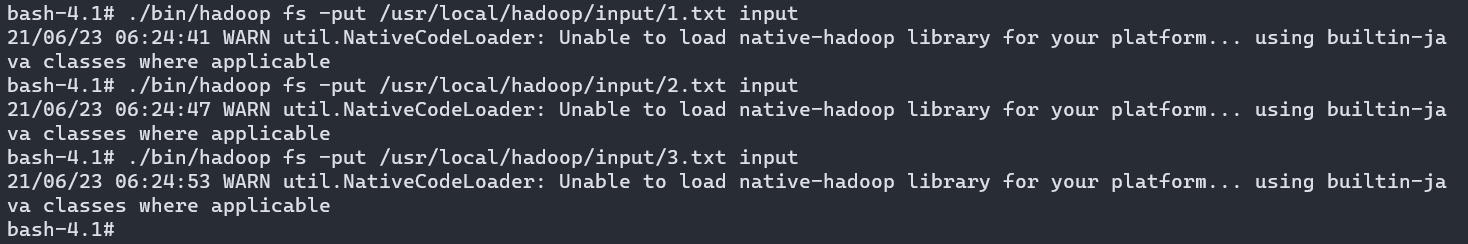
（6）得到结果



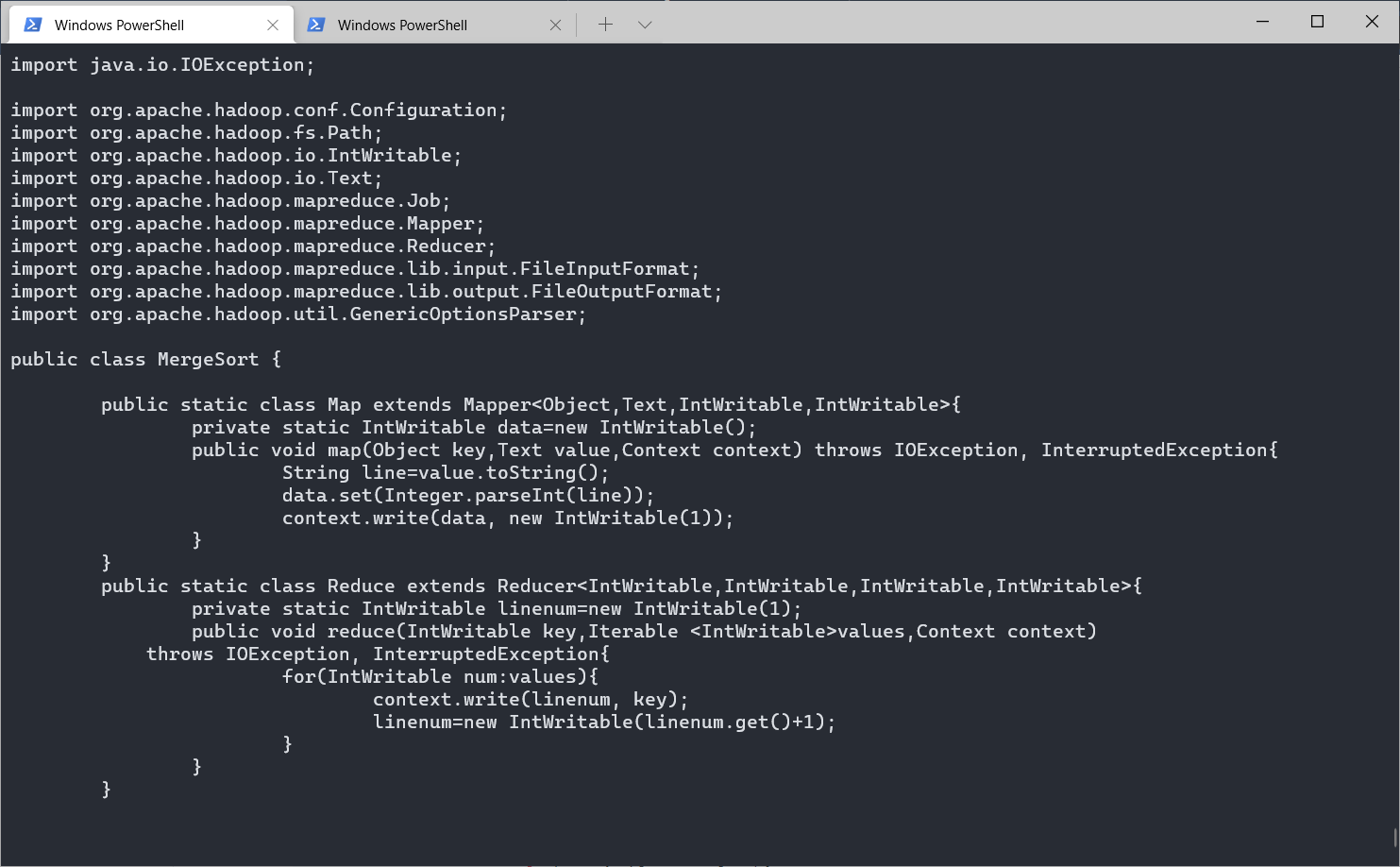
（7）做第二个实验，先删除input output



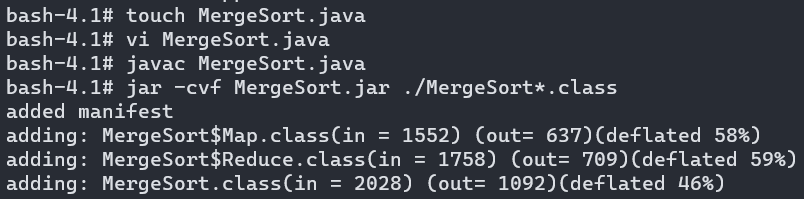
（8）创建文件，并将其上传到HDFS



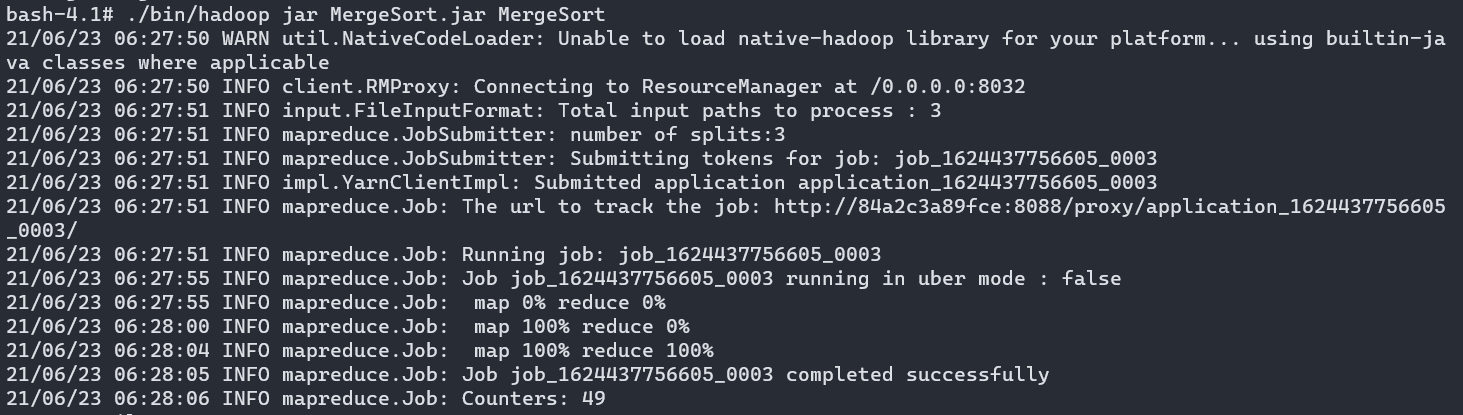
（9）编写代码



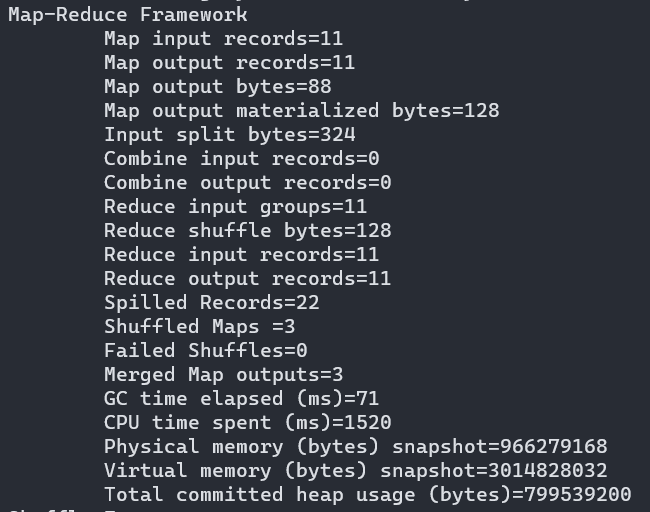
（10）编译代码

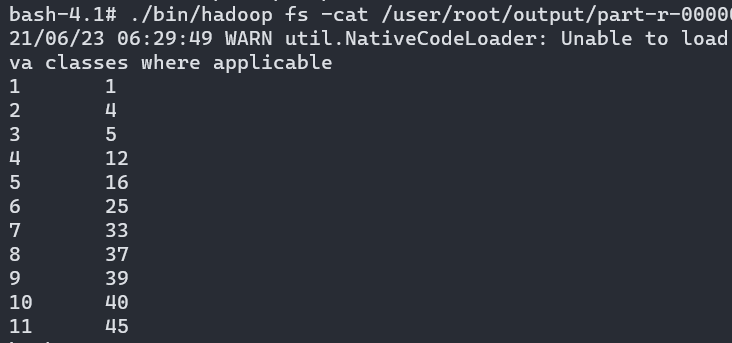


（11）运行jar包

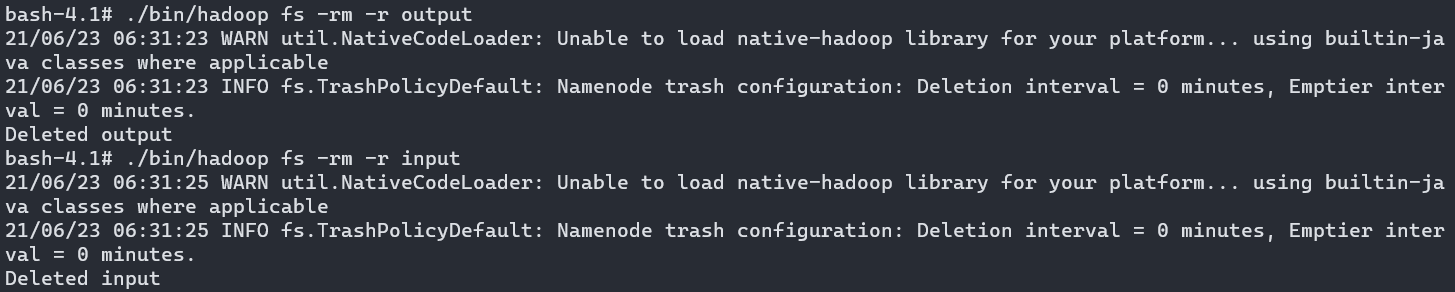


运行成功，结果如下：

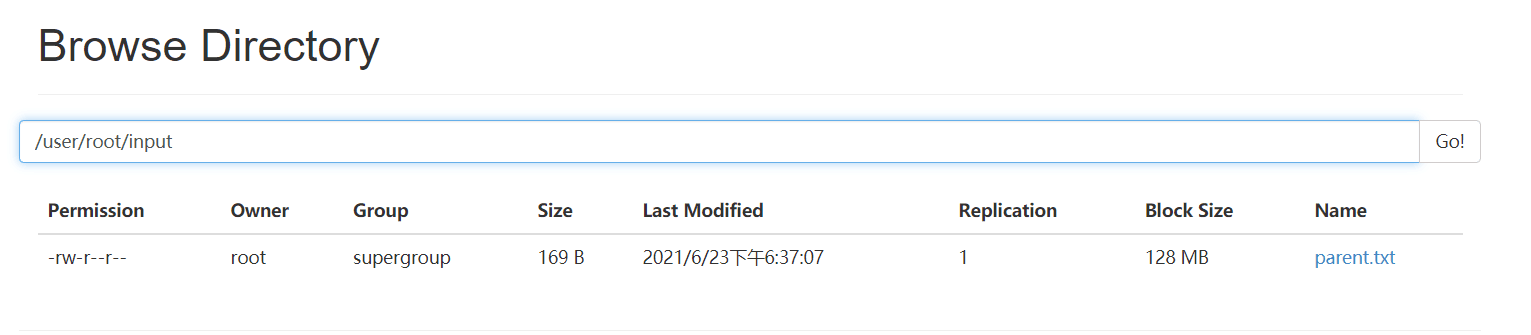




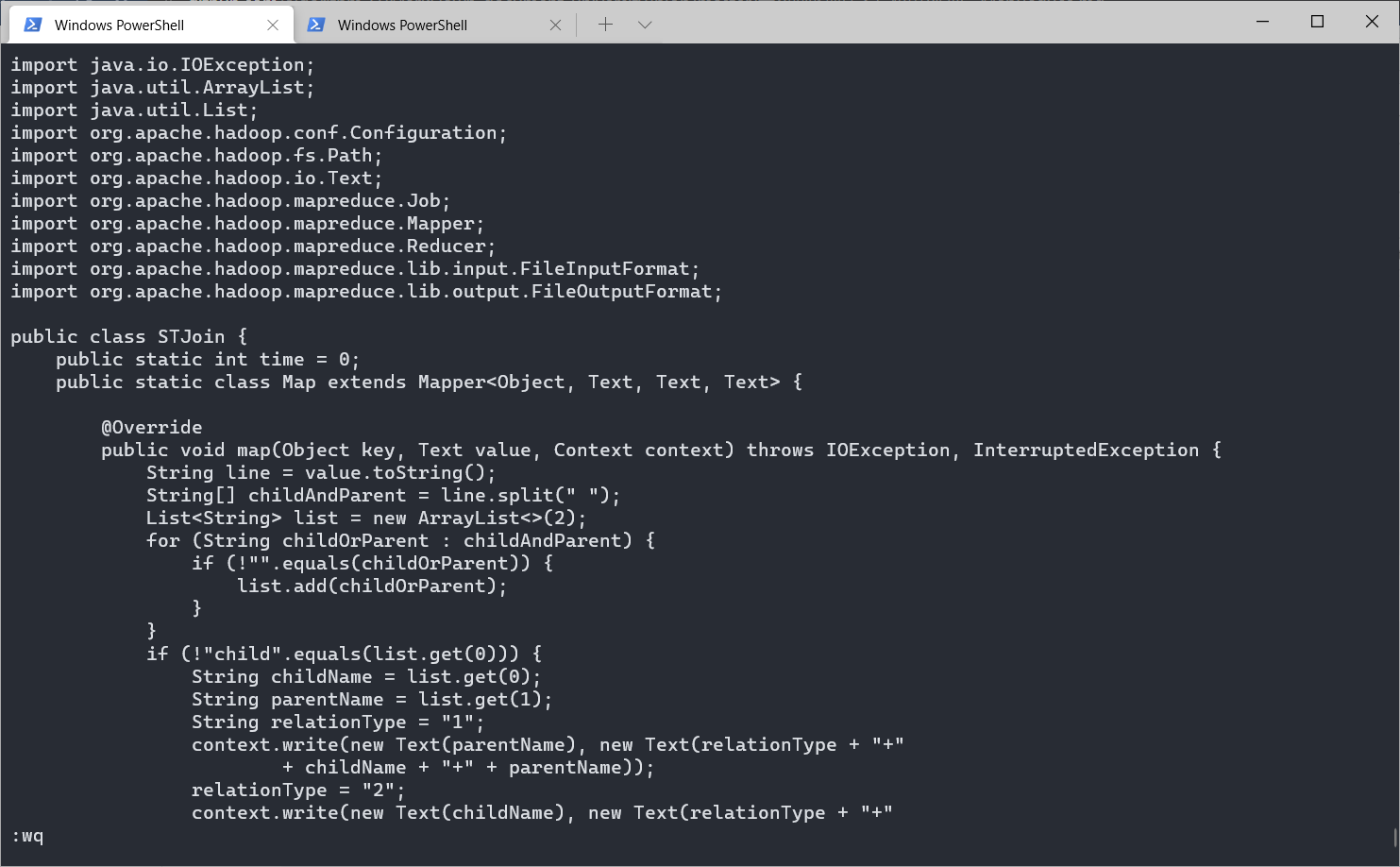
（12）删除input和output文件



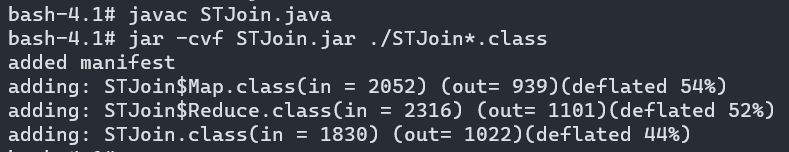
（13）创建新的文本文档并编辑，parent.txt，上传



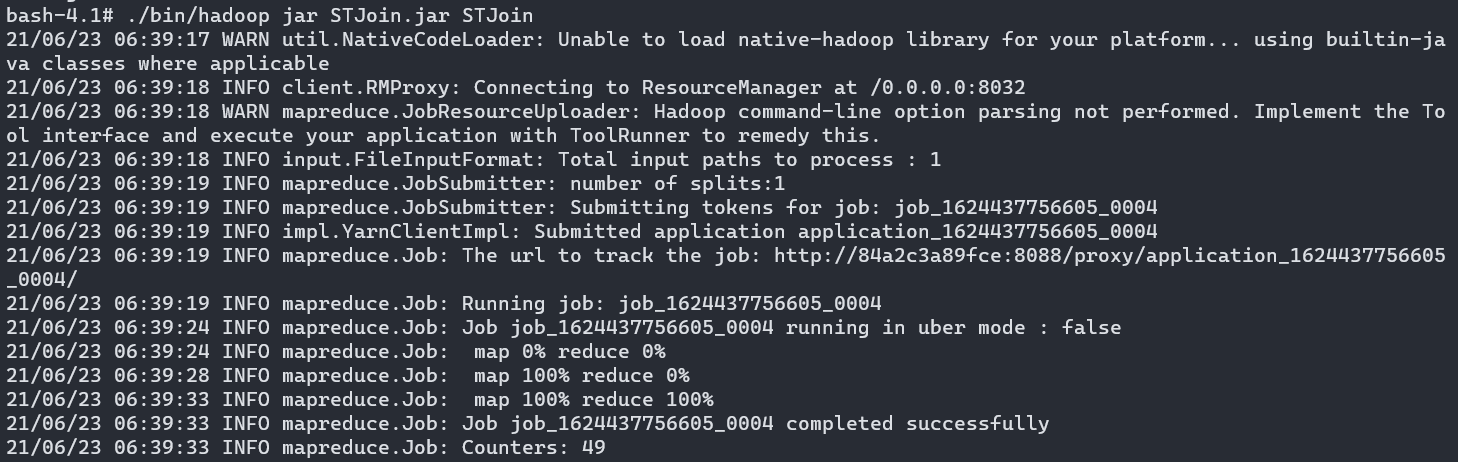
（14）编写代码

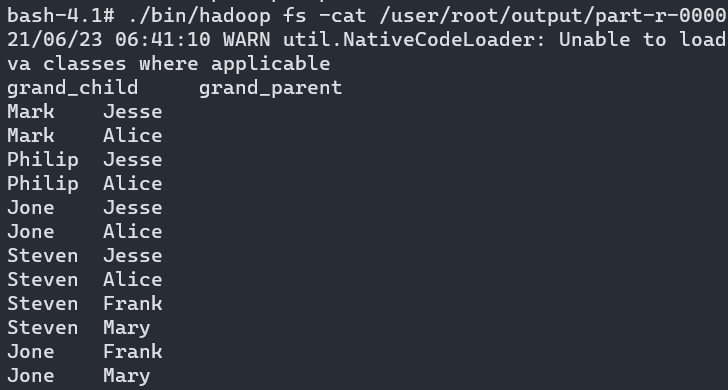


（15）编译代码，生成jar包



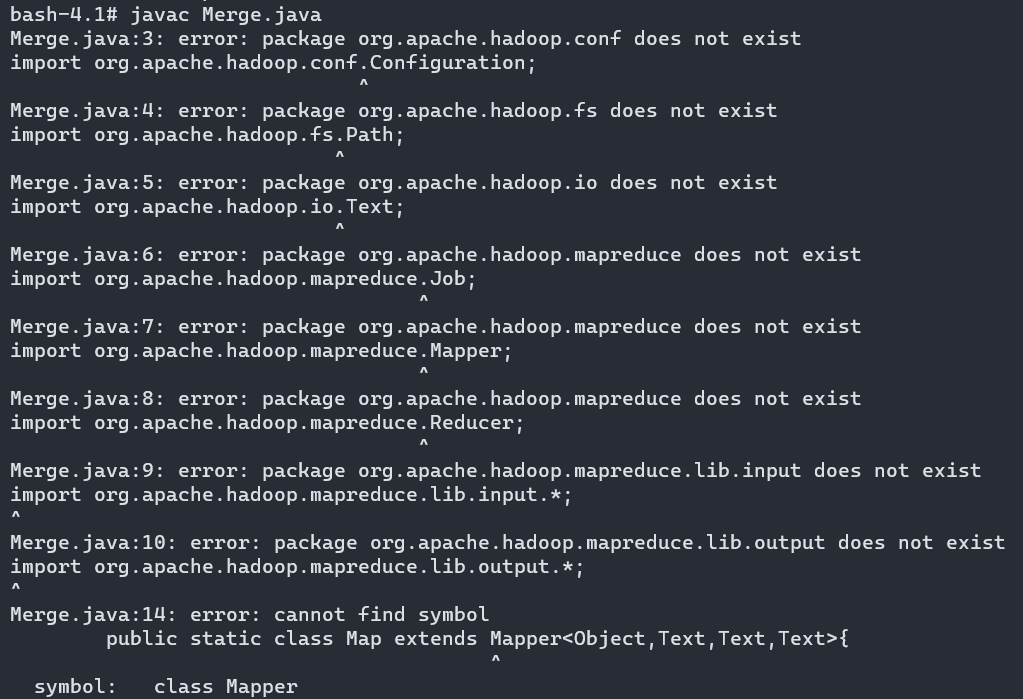
（16）运行jar包，得到结果





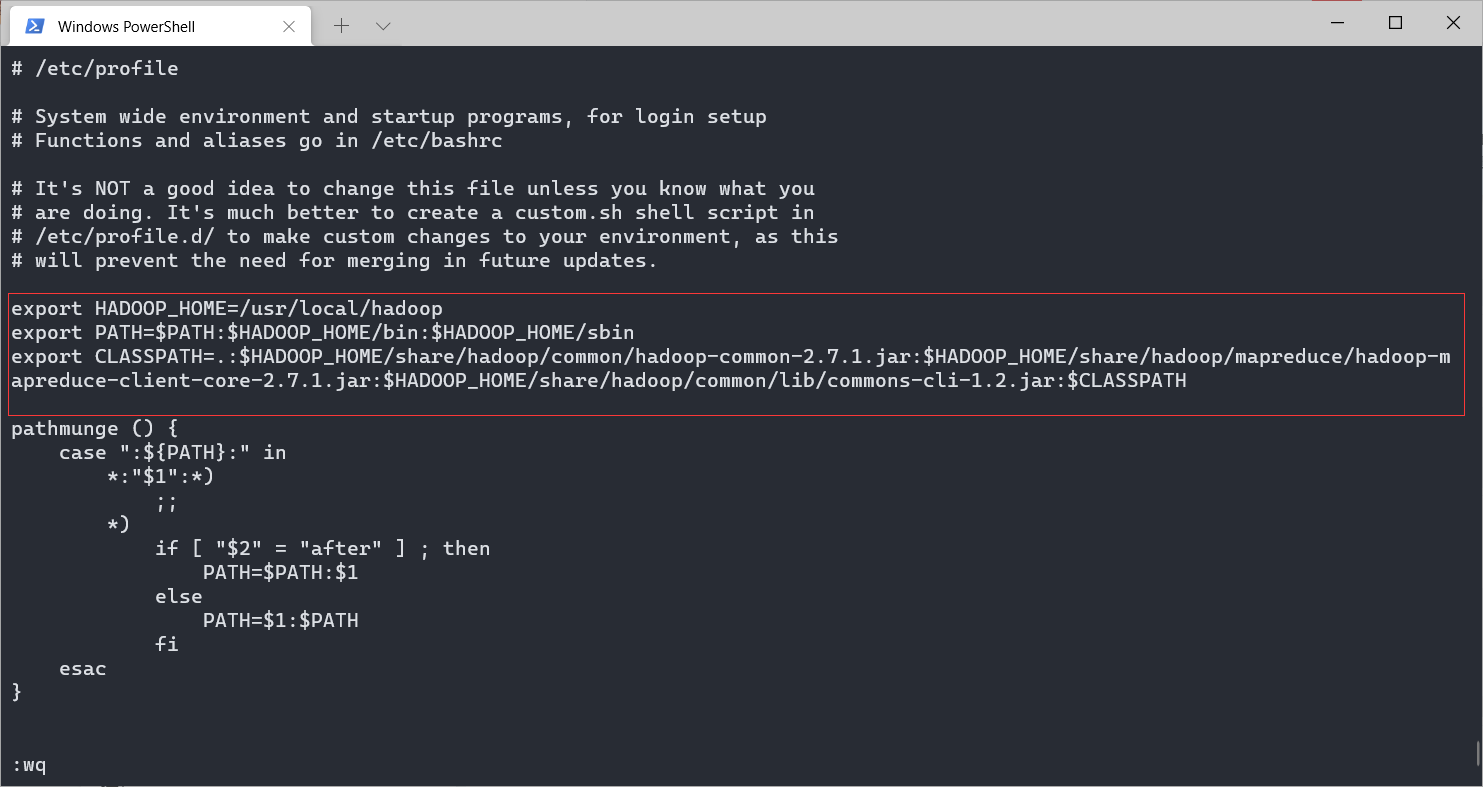
**3.5实验中的问题与解决方法**

（1）编译Java代码时，找不到Hadoop中的Java依赖

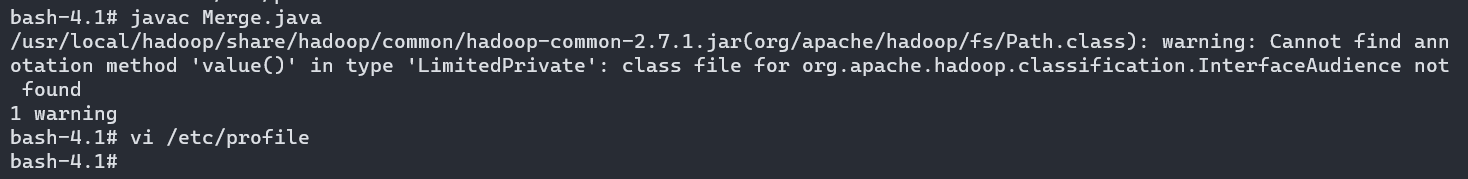


原因：安装Java和Hadoop时没有配置好Hadoop中依赖的路径。

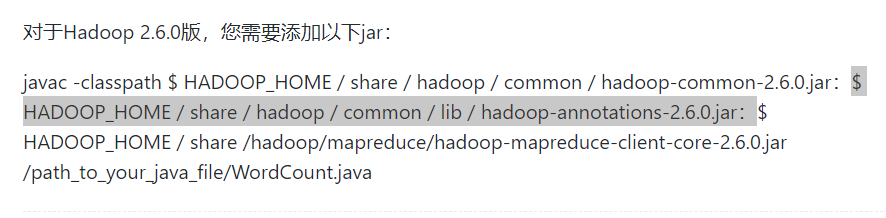
解决方法：编辑/etc/profile文件，添加Hadoop依赖的路径到环境变量。



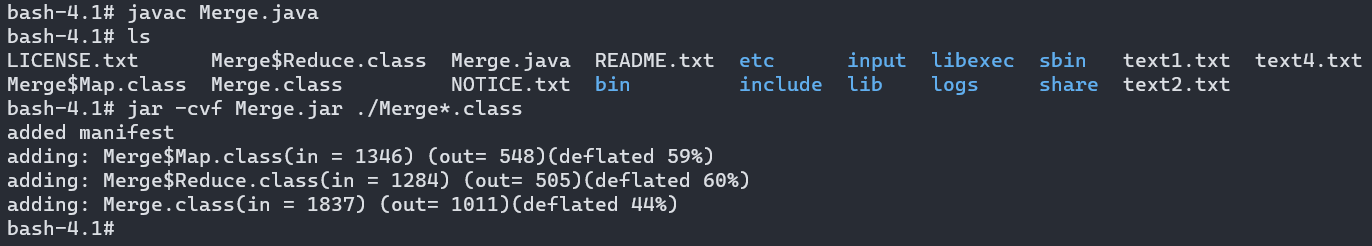
（2）编译时报错，缺少函数“value()”



原因：在某些版本的Hadoop中，需要特别添加hadoop-annotations-2.x.x.jar到环境变量。格式如下图：



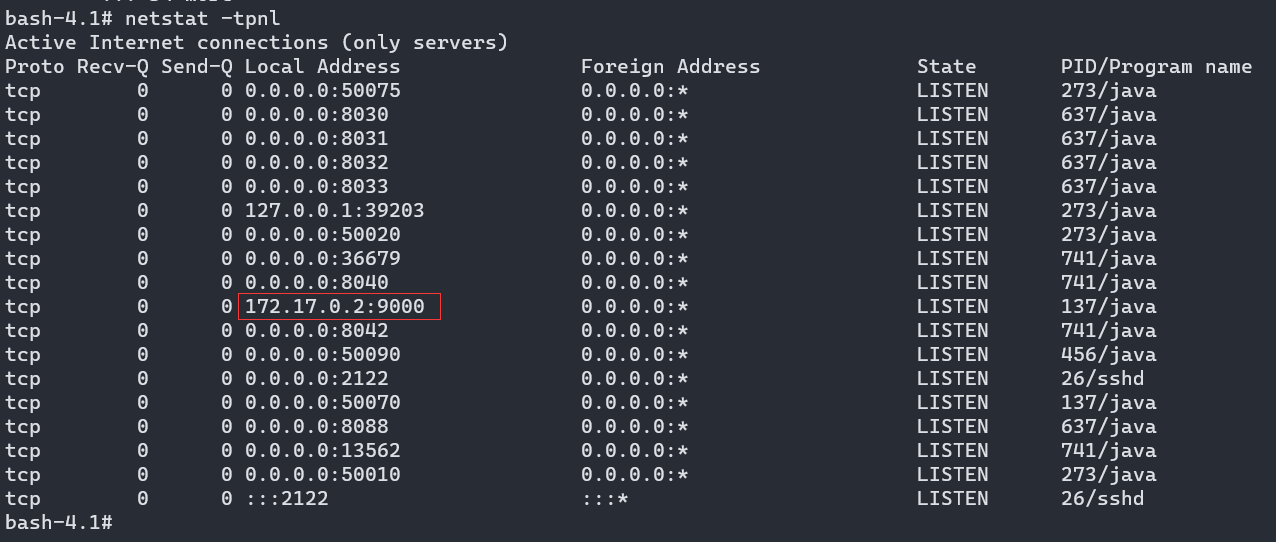
解决方法：添加hadoop-annotations-2.x.x.jar到环境变量，此后编译成功



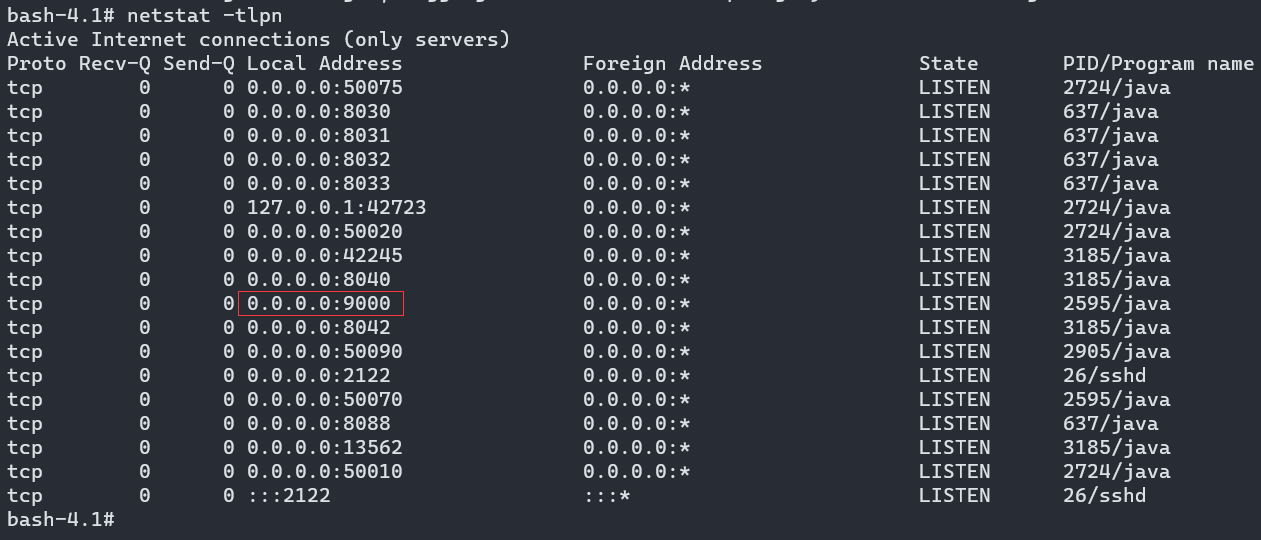
（3）9000端口拒绝了访问



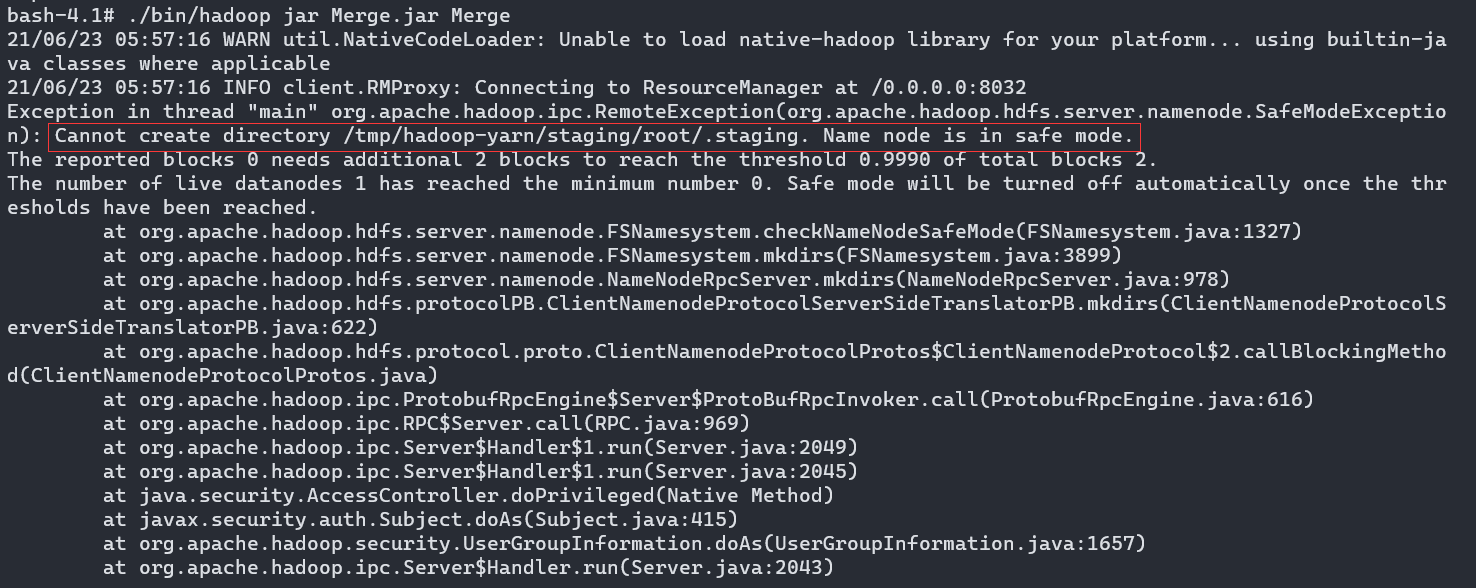
原因：查看网络状态，9000端口已开启，但ip设置成了本机，没有向外界开放。



解决方法：把9000端口修改为向所有人开放，重启Hadoop。

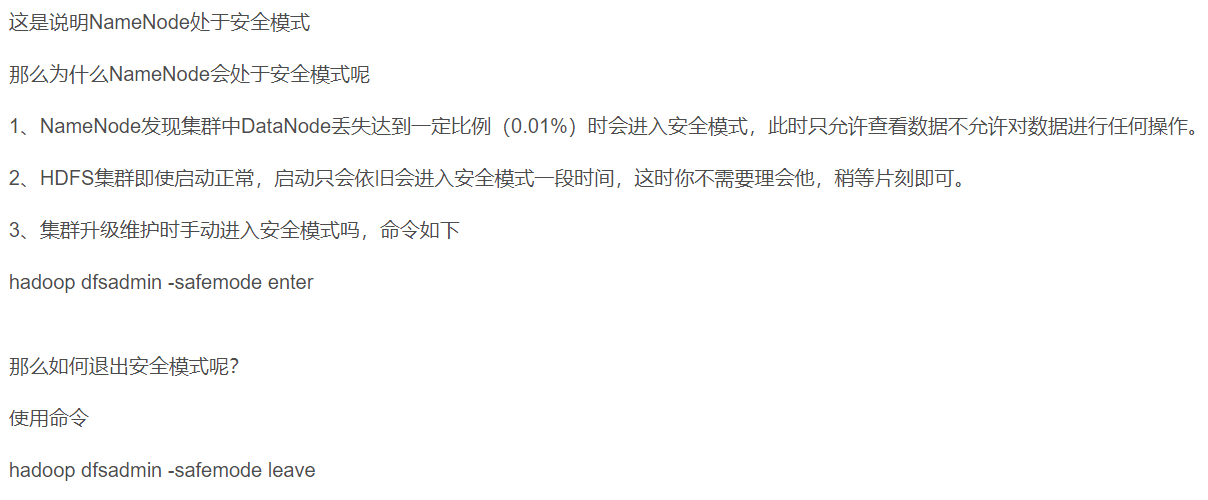


（4）Namenode进入了安全模式，无法运行jar包

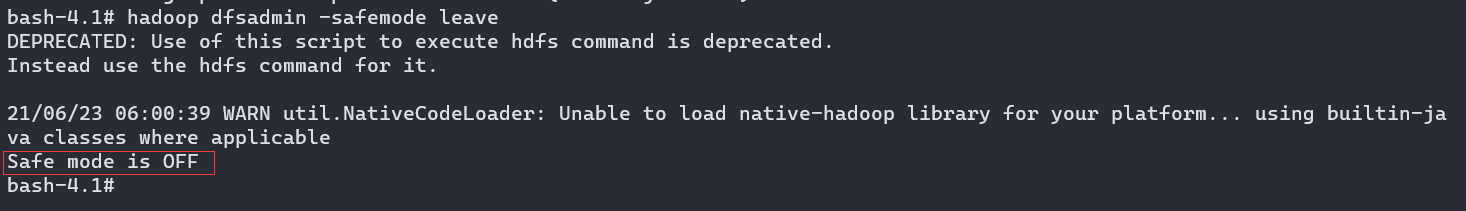


原因：在修改9000端口向所有人开启后，需要重启Hadoop，但此时需要把上次运行时的data删除，Hadoop重启后读取不到原来的文件，认为File Block被破坏，所以自动安全模式。

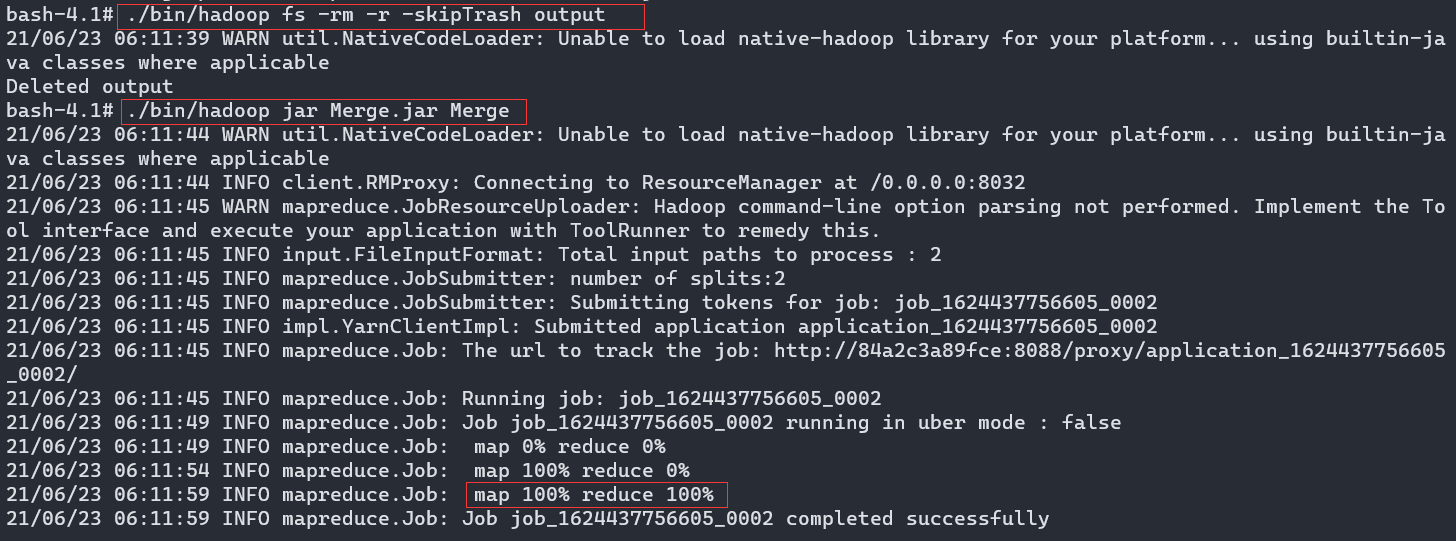




解决方法：关闭安全模式，此时虽然没有了文件，但是HDFS还是对原始数据有记录，所以要把命令行删除原有文件，即使他已经不在了，之后重新上传需要用到的文件即可。



删除原有的input和output文件夹，再次运行，即可编译成功。



第四章 总结与体会

大数据技术是计算机研究领域的一个重要分支，它已经渗透到生活中的各个领域，大数据技术的高速发展为各行业的生命注入了新的血液，给我们的生活带来了极大的便利，这同时对各行业的发展也是一个考验，人们将更加离不开大数据技术，而计算机通过利用海量数据也将更好地服务于人类，使人们的生活更加丰富。未来大数据技术的应用将更加适应人们的生活。

当前，数据科学正在蓬勃发展，研究智能计算的领域十分活跃。虽然目前智能计算和大数据的研究水平暂时还很难使“智能机器”真正具备人类的智能，但大数据技术将在21世纪蓬勃发展，人工智能将不仅是模仿生物脑的功能，而且两者具有相同的特性，这两者的结合将使人工智能的研究向着更广和更深的方向发展，将开辟一个全新的领域，开辟很多新的研究方向。大数据技术将探索智能的新概念、新理论、新方法和新技术，而这些研究将在以后的发展中取得重大的成就。

经过课程设计，使我加深了对所学理论知识的理解与巩固，并能将课程中的纯理论应用到实践中，进一步加深了对知识的认识。同时，也有助于对其他知识的理解。我不但对分布式文件管理有了更深入的理解，还熟练的应用Hadoop、HDFS、Hbase对文件进行各种操作。

第五章 参考与引用

[1] https://blog.csdn.net/liu16659/article/details/80212233

[2] https://blog.csdn.net/ystyaoshengting/article/details/103026872

[3] <https://zhuanlan.zhihu.com/p/269047002>

[4] https://www.shuzhiduo.com/A/GBJrBBQRJ0/

[5] https://stackoverflow.com/questions/48107616/hadoop-blockmissingexception

[6] https://blog.csdn.net/liu16659/article/details/80212233

[7]<https://blog.csdn.net/qq_52679708/article/details/115448087?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_title-0&spm=1001.2101.3001.4242>

**附录一.** **MapReduce编程初级实践实验代码**

1. Merge.java

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class Merge {

public static class Map extends Mapper<Object,Text,Text,Text>{

private static Text text=new Text();

public void map(Object key,Text value,Context context) throws IOException, InterruptedException{

text=value;

context.write(text,new Text(""));

}

}

public static class Reduce extends Reducer<Text,Text,Text,Text>{

public void reduce(Text key,Iterable <Text>values,Context context)

throws IOException, InterruptedException{

context.write(key, new Text(""));

}

}

public static void main(String[] args) throws IOException,

ClassNotFoundException, InterruptedException{

Configuration conf=new Configuration();

conf.set("fs.defaultFS","hdfs://localhost:9000");

String[] otherArgs=new String[]{"input","output"};

if(otherArgs.length!=2){

System.err.println("Usage:Merge and duplicate removal<in><out>");

System.exit(2);

}

Job job=Job.getInstance(conf,"Merge and duplicate removal");

job.setJarByClass(Merge.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job,new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job,new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true)?0:1);

}

}

1. MergeSort.java

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class MergeSort {

public static class Map extends Mapper<Object,Text,IntWritable,IntWritable>{

private static IntWritable data=new IntWritable();

public void map(Object key,Text value,Context context) throws IOException, InterruptedException{

String line=value.toString();

data.set(Integer.parseInt(line));

context.write(data, new IntWritable(1));

}

}

public static class Reduce extends Reducer<IntWritable,IntWritable,IntWritable,IntWritable>{

private static IntWritable linenum=new IntWritable(1);

public void reduce(IntWritable key,Iterable <IntWritable>values,Context context)

throws IOException, InterruptedException{

for(IntWritable num:values){

context.write(linenum, key);

linenum=new IntWritable(linenum.get()+1);

}

}

}

/\*\*

\* @param args

\* @throws IOException

\* @throws InterruptedException

\* @throws ClassNotFoundException

\*/

public static void main(String[] args) throws IOException,

ClassNotFoundException, InterruptedException{

Configuration conf=new Configuration();

conf.set("fs.defaultFS","hdfs://localhost:9000");

String[] str=new String[]{"input","output"};

String[] otherArgs=new GenericOptionsParser(conf,str).getRemainingArgs();

if(otherArgs.length!=2){

System.err.println("Usage:mergesort<in><out>");

System.exit(2);

}

Job job=Job.getInstance(conf,"mergesort");

job.setJarByClass(MergeSort.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job,new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job,new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true)?0:1);

}

}

1. STJoin.java

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class STJoin {

public static int time = 0;

public static class Map extends Mapper<Object, Text, Text, Text> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String[] childAndParent = line.split(" ");

List<String> list = new ArrayList<>(2);

for (String childOrParent : childAndParent) {

if (!"".equals(childOrParent)) {

list.add(childOrParent);

}

}

if (!"child".equals(list.get(0))) {

String childName = list.get(0);

String parentName = list.get(1);

String relationType = "1";

context.write(new Text(parentName), new Text(relationType + "+"

+ childName + "+" + parentName));

relationType = "2";

context.write(new Text(childName), new Text(relationType + "+"

+ childName + "+" + parentName));

}

}

}

public static class Reduce extends Reducer<Text, Text, Text, Text> {

@Override

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

if (time == 0) {

context.write(new Text("grand\_child"), new Text("grand\_parent"));

time++;

}

List<String> grandChild = new ArrayList<>();

List<String> grandParent = new ArrayList<>();

for (Text text : values) {

String s = text.toString();

String[] relation = s.split("\\+");

String relationType = relation[0];

String childName = relation[1];

String parentName = relation[2];

if ("1".equals(relationType)) {

grandChild.add(childName);

} else {

grandParent.add(parentName);

}

}

int grandParentNum = grandParent.size();

int grandChildNum = grandChild.size();

if (grandParentNum != 0 && grandChildNum != 0) {

for (int m = 0; m < grandChildNum; m++) {

for (int n = 0; n < grandParentNum; n++) {

context.write(new Text(grandChild.get(m)), new Text(

grandParent.get(n)));

}

}

}

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://localhost:9000");

String[] otherArgs = new String[]{"input", "output"};

if (otherArgs.length != 2) {

System.err.println("Usage: Single Table Join <in> <out>");

System.exit(2);

}

Job job = Job.getInstance(conf, "Single table Join ");

job.setJarByClass(STJoin.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}