

# 暨南大学本科实验报告专用纸

课程名称 云计算实验 成绩评定 \_\_\_\_\_  
实验项目名称 分布式文件系统 指导教师 魏林锋  
实验项目编号 08060309 实验项目类型 验证 实验地点 线上  
学生姓名 陈宇 学号 2020101642  
学院 信息科学技术学院 系 计算机系 专业 软件工程  
实验时间 2022 年 9 月 27 日 上午 ~ 9 月 27 日 下午 温度    °C 湿度   

## 1.1. 实验目的

- 1) 了解 Hadoop Shell 命令的用法。
- 2) 熟悉 HDFS 中的 JAVA API 的使用。
- 3) 通过实验掌握基本的 HDFS 中的 JAVA API 编程方法。
- 4) 掌握 hadoop 分布式文件系统 Shell 命令的基本操作。

## 1.2. 实验原理

### HDFS Java API 介绍:

由于 Hadoop 是由 Java 语言编写的, 因此可以使用 java API 操作 Hadoop 文件系统。

HDFS Shell 本质上就是对 java API 的应用, 通过编程的形式操作 HDFS, 其核心是使用

HDFS 提供的 Java API 构造一个客户端对象, 然后通过客户端对 HDFS 上的文件进行操作

(增, 删, 改, 查)

包名	功能描述
org.apache.hadoop.fs.FileSystem	它是通用文件系统的抽象基类，可以被分布式文件系统继承，它具有许多实现类
org.apache.hadoop.fs.FileStatus	它用于向客户端展示系统中文件和目录的元数据
org.apache.hadoop.fs.FSDataInputStream	文件输入流，用于读取Hadoop文件
org.apache.hadoop.fs.FSDataOutputStream	文件输出流，用于写Hadoop文件
org.apache.hadoop.conf.Configuration	访问配置项，默认配置参数在core-site.xml中
org.apache.hadoop.fs.Path	表示Hadoop文件系统中的文件或者目录的路径

## HDFS Shell 命令使用

- 创建文件夹
- 删除目录和文件
- 上传/下载
- 查看文件内容
- 追写文件
- 查看磁盘占用空间

## HDFS Java API 编程

- Eclipse 插件开发配置
- 上传文件编程
- 创建文件编程
- 查找文件信息编程
- 从 HDFS 下载文件到本地编程

## 1.3. 实验环境

已经配置完成的 Hadoop 伪分布式或完全分布式环境。

我选择已经配置完的完全分布式环境

机器名称	IP 地址
Master	172.17.0.2
Slave1	172.17.0.3
Slave2	172.17.0.4

## 1.4. 实验内容和过程

### 1.4.1. HDFS Shell 命令使用

hadoop fs 与 hdfs dfs 的命令的使用是相似的，本实验使用的是 hdfs dfs 命令，所有命令的操作都是在 hadoop 用户下进行。

- **1.4.1.1. mkdir: 创建文件夹**

使用方法: `hdfs fs -mkdir [-p] <paths>`

接受路径指定的 uri 作为参数，创建这些目录。其行为类似于 Unix 的 `mkdir -p`，它会创建路径中的各级父目录。

**示例 1:**在分布式主目录下新建文件夹 test

```
[hadoop@master ~]$ hdfs dfs -mkdir /test
```

```
[hadoop@master ~]$ hdfs dfs -ls /
```

```
[hadoop@master ~]$ hdfs dfs -ls /  
Found 1 items  
drwxr-xr-x  - hadoop supergroup          0 2022-09-28 02:17 /test  
[hadoop@master ~]$
```

**示例 2:** 在根目录下新建文件夹/test/dir0/dir1,如果上一级目录不存在, 需要使用 -p 参数。

```
[hadoop@master ~]$ hdfs dfs -mkdir -p /test/dir0/dir1
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/dir0  
Found 1 items  
drwxr-xr-x  - hadoop supergroup          0 2022-09-28 07:43 /test/dir0/dir1  
[hadoop@master ~]$
```

- **1.4.1.2. touch:新建文件**

使用方法: `hdfs fs -touchz URI [URI ...]`

当前时间下创建大小为 0 的空文件, 若大小不为 0, 返回错误信息。

**示例:** 在/test/下新建文件 file1

```
[hadoop@master ~]$ hdfs dfs -touchz /test/file1
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/
```

```
[hadoop@master ~]$ hdfs dfs -touchz /test/file1  
[hadoop@master ~]$ hdfs dfs -ls /test/  
Found 2 items  
drwxr-xr-x  - hadoop supergroup          0 2022-09-28 07:43 /test/dir0  
-rw-r--r--  3 hadoop supergroup          0 2022-09-28 07:46 /test/file1  
[hadoop@master ~]$
```

- **1.4.1.3. ls 列指定目录文件和目录**

使用方法: `hdfs dfs -ls [-d][-h][-R]<paths>`

表 1: ls 命令选项和功能

选项	功能说明
-d	返回 paths
-h	按照 KMG 数据大小单位显示文件大小, 如果没有单位, 默认认为 B
-R	级联显示 paths 下文件, 这里 paths 是个 多级目录

**示例 1:** 列出/test 目录下的所有文件和目录信息

```
[hadoop@master ~]$ hdfs dfs -ls /test
```

```
[hadoop@master ~]$ hdfs dfs -ls /test
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2022-09-28 07:43 /test/dir0
-rw-r--r--   3 hadoop supergroup          0 2022-09-28 07:46 /test/file1
[hadoop@master ~]$
```

**示例 2:** 列出/test 目录信息

```
[hadoop@master ~]$ hdfs dfs -ls -d /test
```

```
[hadoop@master ~]$ hdfs dfs -ls -d /test
drwxr-xr-x   - hadoop supergroup          0 2022-09-28 07:46 /test
[hadoop@master ~]$
```

**示例 3:** 列出目录和文件的大小

```
[hadoop@master ~]$ hdfs dfs -ls -h /test
```

因为这些文件和目录都是新建的, 所以文件和目录的大小为零。

```
[hadoop@master ~]$ hdfs dfs -ls -h /test
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2022-09-28 07:43 /test/dir0
-rw-r--r--   3 hadoop supergroup          0 2022-09-28 07:46 /test/file1
[hadoop@master ~]$
```

**示例 4：**循环列出目录、子目录及文件信息

```
[hadoop@master ~]$ hdfs dfs -ls -R /test
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test
drwxr-xr-x   - hadoop supergroup          0 2022-09-28 07:43 /test/dir0
drwxr-xr-x   - hadoop supergroup          0 2022-09-28 07:43 /test/dir0/dir1
-rw-r--r--   3 hadoop supergroup          0 2022-09-28 07:46 /test/file1
[hadoop@master ~]$
```

- **1.4.1.4. rm 删除目录和文件**

使用方法：hdfs dfs -rm [-f] [-r|-R] [-skip Trash] <paths>

表二 rm 命令的选项和功能

选项	说明
-f	如果要删除的文件不存在, 不显示提示 和错误信息
-r R	级联删除目录下的所有文件和子目录文件
-skipTrash	直接删除, 不进入垃圾回收站

**示例 1：**删除文件

```
[hadoop@master ~]$ hdfs dfs -touchz /test/dir0/file3
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -rm /test/dir0/file3
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
```

结果： 删除前

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
drwxr-xr-x  - hadoop supergroup          0 2022-09-28 07:43 /test/dir0/dir1
-rw-r--r--  3 hadoop supergroup          0 2022-09-28 08:07 /test/dir0/file3
[hadoop@master ~]$
```

删除后

```
[hadoop@master ~]$ hdfs dfs -rm /test/dir0/file3
Deleted /test/dir0/file3
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
drwxr-xr-x  - hadoop supergroup          0 2022-09-28 07:43 /test/dir0/dir1
[hadoop@master ~]$
```

示例 2： 删除目录及目录下的目录和文件

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -rm -r /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/
```

结果：

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
drwxr-xr-x  - hadoop supergroup          0 2022-09-28 07:43 /test/dir0/dir1
[hadoop@master ~]$ hdfs dfs -rm -r /test/dir0
Deleted /test/dir0
[hadoop@master ~]$ hdfs dfs -ls -R /test/
-rw-r--r--  3 hadoop supergroup          0 2022-09-28 07:46 /test/file1
[hadoop@master ~]$
```

#### ● 1.4.1.5. put/get:上传/下载

使用方法： `hdfs dfs -put [-f] [-p] <localsrc>...<dst>`

`hdfs dfs -get [-p] [-ignoreCrc] [-crc] <src>...<localdst>`

put	将本地文件系统的复制到 HDFS 文件系统的目录下
get	将 HDFS 中的文件复制到本地文件系统

	统中，与-put 命令相反
-f	如果文件在分布式文件系统上已经存在，则覆盖存储，若不加则会报错；
-p	保持源文件的属性（组、拥有者、创建时间、权限等）；
-ignoreCrc	同上。

**示例 1：**把本地新建的文件 test.txt 放到分布式文件系统主目录下，保存名为 hfile;

```
[hadoop@master ~]$ touch /tmp/test.txt
```

```
[hadoop@master ~]$ ls -l /tmp/test.txt
```

```
[hadoop@master ~]$ touch /tmp/test.txt
[hadoop@master ~]$ ls -l /tmp/test.txt
ls: cannot access /tmp/test.txt: No such file or directory
[hadoop@master ~]$ ls -l /tmp/test.txt
-rw-rw-r--. 1 hadoop hadoop 0 Sep 28 08:25 /tmp/test.txt
[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -put /tmp/test.txt /test/hfile
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/
```

```
[hadoop@master ~]$ hdfs dfs -put /tmp/test.txt /test/hfile
[hadoop@master ~]$ hdfs dfs -ls /test/
Found 2 items
-rw-r--r--  3 hadoop supergroup      0 2022-09-28 07:46 /test/file1
-rw-r--r--  3 hadoop supergroup      0 2022-09-28 08:27 /test/hfile
[hadoop@master ~]$
```

**示例 2：**把分布式文件系统目录下的文件复制到本地

```
[hadoop@master3 hadoop]$ hdfs dfs -get /test/hfile /home/hadoop/hfile
```

```
[hadoop@master ~]$ ls -l /home/hadoop/hfile
```



```
[hadoop@master ~]$ hdfs dfs -get /test/hfile /home/hadoop/hfile
[hadoop@master ~]$ ls -l /home/hadoop/hfile
-rw-r--r--. 1 hadoop hadoop 0 Sep 28 08:29 /home/hadoop/hfile
[hadoop@master ~]$
```

**示例 3:** 把本地新建的文件 test.txt 放到分布式文件系统主目录下, 覆盖原来的文件

```
[hadoop@master ~]$ hdfs dfs -ls /test/hfile
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/hfile
-rw-r--r-- 3 hadoop supergroup 0 2022-09-28 08:27 /test/hfile
[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -put -f /home/hadoop/hfile /test/hfile
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/hfile
```

```
[hadoop@master ~]$ hdfs dfs -put -f /home/hadoop/hfile /test/hfile
[hadoop@master ~]$ hdfs dfs -ls /test/hfile
-rw-r--r-- 3 hadoop supergroup 0 2022-09-28 08:32 /test/hfile
[hadoop@master ~]$
```

**示例 4:** 把本地新建的文件 test.txt 放到分布式文件系统主目录下, 保持源文件属性

```
[hadoop@master ~]$ ls -l /home/hadoop/file
ls: cannot access /home/hadoop/file: No such file or directory
[hadoop@master ~]$ touch /home/hadoop/file
[hadoop@master ~]$ ls -l /home/hadoop/file
-rw-rw-r--. 1 hadoop hadoop 0 Sep 28 08:36 /home/hadoop/file
[hadoop@master ~]$
```

```
[hadoop@master ~]$ ls -l /home/hadoop/file
```

```
[hadoop@master ~]$ ls -l /home/hadoop/file
-rw-rw-r--. 1 hadoop hadoop 0 Sep 28 08:36 /home/hadoop/file
[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -put -p /home/hadoop/file /test/
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/file
```

```
[hadoop@master ~]$ hdfs dfs -put -p /home/hadoop/file /test/
[hadoop@master ~]$ hdfs dfs -ls /test/file
-rw-rw-r-- 3 hadoop hadoop 0 2022-09-28 08:36 /test/file
[hadoop@master ~]$
```

- 1.4.1.6. cat、text、tail:查看文件内容

使用方法: `hdfs dfs -cat/text [-ignoreCrc] <src>`

`Hdfs dfs -tail [-f] <file>`

其中:

-ignoreCrc	忽循环检验失败的文件;
-f	动态更新显示数据, 如查看某个不 断增长的文件的日志文件。

3 个命令都是在命令行窗口查看指定文件内容。区别是 `text` 不仅可以查看文 本文件, 还可以查看压缩文件和 `Avro` 序列化的文件, 其他两个不可以; `tail` 查看 的是最后 1KB 的文件 (Linux 上的 `tail` 默认查看最后 10 行记录)。

示例 1: 查看文件的内容

```
[hadoop@master ~]$ hdfs dfs -cat /test/file
```

```
[hadoop@master ~]$ hdfs dfs -cat /test/file
hello world !!!
im from jnu    !
i am a student!!!
i like watching movies and jogging  !!

[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -text /test/file
```

```
[hadoop@master ~]$ hdfs dfs -text /test/file
hello world !!!
im from jnu !
i am a student!!!
i like watching movies and jogging !!

[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -tail /test/file
```

```
[hadoop@master ~]$ hdfs dfs -tail /test/file
hello world !!!
im from jnu !
i am a student!!!
i like watching movies and jogging !!

[hadoop@master ~]$
```

- **1.4.1.7. appendToFile:追写文件**

使用方法: `hdfs dfs -appendToFile <localsrc>...<dst>`

**示例 1.**把本地文件系统文件追加到分布式文件系统中

```
[hadoop@master ~]$ cat /home/hadoop/file2
```

```
[hadoop@master ~]$ hdfs dfs -appendToFile /home/hadoop/file2 /test/file2
```

```
[hadoop@master ~]$ hdfs dfs -cat
```

```
[hadoop@master ~]$ ls /home/hadoop/
file hfile
[hadoop@master ~]$ touch /home/hadoop/file2
[hadoop@master ~]$ vi /home/hadoop/file2
[hadoop@master ~]$ cat /home/hadoop/file2
Hello HDFS!
[hadoop@master ~]$ hdfs dfs -appendToFile /home/hadoop/file2 /test/file2
[hadoop@master ~]$ hdfs dfs -cat /test/file2
Hello HDFS!
[hadoop@master ~]$
```

- **1.4.1.8. du: 显示占用磁盘空间大小**

使用方法： `hdfs dfs -du [-s] [-h] <path>...`

默认按字节显示指定目录所占空间大小。

-s	显示指定目录下文件总大小；
-h	按照 KMG 数据大小单位显示文件大小，如果没有单位，默认为 B。

**示例 1：** 显示分布式主目录下文件和目录大小

`[hadoop@master ~]$ hdfs dfs -du /test/`

```
[hadoop@master ~]$ hdfs dfs -du /test/
94 282 /test/file
0 0 /test/file1
12 36 /test/file2
0 0 /test/hfile
[hadoop@master ~]$
```

`[hadoop@master ~]$ hdfs dfs -du -h /test/`

```
[hadoop@master ~]$ hdfs dfs -du -h /test/
94 282 /test/file
0 0 /test/file1
12 36 /test/file2
0 0 /test/hfile
[hadoop@master ~]$
```

`[hadoop@master ~]$ hdfs dfs -du -s /test/`

```
[hadoop@master ~]$ hdfs dfs -du -s /test/
106 318 /test
[hadoop@master ~]$
```

- **1.4.1.9. cp 复制文件** 将文件从 SRC 复制到 DST，如果指定了多个 SRC，则 DST 必须为一个目录

使用方法： `hdfs fs -cp SRC [SRC ...] DST`

### 示例 1

```
[hadoop@master ~]$ hdfs dfs -mkdir /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -cp /test/file2 /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/dir0
```

```
Found 1 items
-rw-r--r--  3 hadoop supergroup      12 2022-09-28 09:14 /test/dir0/file2
[hadoop@master ~]$
```

- **1.4.1.10. chmod 修改文件权限**

使用方法: `hdfs fs -chmod [-R] <MODE[,MODE]... | OCTALMODE>URI [URI ...]`

改变文件的权限。使用-R 将使改变在目录结构下递归进行。命令的使用者必须是文件的所有者或者超级用户。更多的信息请参见 HDFS 权限用户指南。

### 示例 1:

```
[hadoop@master ~]$ hdfs dfs -chmod 777 /test/file2
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/file2
```

```
[hadoop@master ~]$ hdfs dfs -ls /test
Found 5 items
drwxr-xr-x  - hadoop supergroup      0 2022-09-28 09:14 /test/dir0
-rw-rw-r--  3 hadoop hadoop          94 2022-09-28 08:52 /test/file
-rw-r--r--  3 hadoop supergroup      0 2022-09-28 07:46 /test/file1
-rw-r--r--  3 hadoop supergroup     12 2022-09-28 09:04 /test/file2
-rw-r--r--  3 hadoop supergroup      0 2022-09-28 08:32 /test/hfile
[hadoop@master ~]$ hdfs dfs -chmod 777 /test/file2
[hadoop@master ~]$ hdfs dfs -ls /test/file2
-rwxrwxrwx  3 hadoop supergroup     12 2022-09-28 09:04 /test/file2
[hadoop@master ~]$
```

### 示例 2:

```
[hadoop@master ~]$ hdfs dfs -chmod -R 777 /test/dir0/
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/
Found 5 items
drwxr-xr-x   - hadoop supergroup          0 2022-09-28 09:14 /test/dir0
-rw-rw-r--   3 hadoop hadoop              94 2022-09-28 08:52 /test/file
-rw-r--r--   3 hadoop supergroup          0 2022-09-28 07:46 /test/file1
-rwxrwxrwx   3 hadoop supergroup         12 2022-09-28 09:04 /test/file2
-rw-r--r--   3 hadoop supergroup          0 2022-09-28 08:32 /test/hfile
[hadoop@master ~]$ hdfs dfs -chmod -R 777 /test/dir0/
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
-rwxrwxrwx   3 hadoop supergroup         12 2022-09-28 09:14 /test/dir0/file2
[hadoop@master ~]$
```

#### ● 1.4.1.11. chown 修改文件属主和组

使用方法：hdfsdfs -chmod [-R] <MODE[,MODE]...[OCTALMODE]>URI [URI ...]

改变文件的权限。使用-R 将使改变在目录结构下递归进行。命令的使用者必须是文件的所有者或者超级用户。更多的信息请参见 HDFS 权限用户指南。

##### 示例 1:

```
[hadoop@master ~]$ hdfs dfs -chown hadoop:hadoop /test/file2
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/file2
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/file2
-rwxrwxrwx   3 hadoop supergroup         12 2022-09-28 09:04 /test/file2
[hadoop@master ~]$ hdfs dfs -chown hadoop:hadoop /test/file2
[hadoop@master ~]$ hdfs dfs -ls /test/file2
-rwxrwxrwx   3 hadoop hadoop             12 2022-09-28 09:04 /test/file2
[hadoop@master ~]$
```

##### 示例 2:

```
[hadoop@master ~]$ hdfs dfs -chown -R hadoop:hadoop /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/dir0
Found 1 items
-rwxrwxrwx   3 hadoop supergroup      12 2022-09-28 09:14 /test/dir0/file2
[hadoop@master ~]$ hdfs dfs -chown -R hadoop:hadoop /test/dir0
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir0
-rwxrwxrwx   3 hadoop hadoop          12 2022-09-28 09:14 /test/dir0/file2
[hadoop@master ~]$
```

- **1.4.1.12. chgrp 修改文件属组**

使用方法: `hdfs fs -chgrp [-R] GROUP URI [URI ...]`

改变文件所属的组。使用-R 将使改变在目录结构下递归进行。命令的使用者必须是文件的所有者或者超级用户。更多的信息请参见 HDFS 权限用户指南。

#### 示例 1

```
[hadoop@master ~]$ hdfs dfs -ls /test/file5
```

```
[hadoop@master ~]$ hdfs dfs -chgrp hadoop /test/file5
```

```
[hadoop@master ~]$ hdfs dfs -ls /test/file5
```

```
[hadoop@master ~]$ hdfs dfs -chgrp hadoop /test/file5
[hadoop@master ~]$ hdfs dfs -ls /test/file5
-rw-r--r--   3 hadoop hadoop          0 2022-09-28 12:02 /test/file5
[hadoop@master ~]$
```

#### 示例 2

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir1
```

```
[hadoop@master ~]$ hdfs dfs -chgrp -R hadoop /test/dir1
```

```
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir1
```

```
[hadoop@master ~]$ hdfs dfs -mkdir /test/dir1
[hadoop@master ~]$ hdfs dfs -touchz /test/dir1/file6
[hadoop@master ~]$ hdfs dfs -touchz /test/dir1/file7
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir1
-rw-r--r--  3 hadoop supergroup      0 2022-09-28 12:05 /test/dir1/file6
-rw-r--r--  3 hadoop supergroup      0 2022-09-28 12:05 /test/dir1/file7
[hadoop@master ~]$ hdfs dfs -chgrp -R hadoop /test/dir1
[hadoop@master ~]$ hdfs dfs -ls -R /test/dir1
-rw-r--r--  3 hadoop hadoop          0 2022-09-28 12:05 /test/dir1/file6
-rw-r--r--  3 hadoop hadoop          0 2022-09-28 12:05 /test/dir1/file7
[hadoop@master ~]$
```

- **1.4.1.13. dfsadmin 显示 HDFS 运行状态和管理 HDFS**

dfsadmin 是一个多任务客服端工具，用来显示 HDFS 运行状态和管理 HDFS，文件的命

令

```
[hadoop@master ~]$ hdfs dfsadmin -help
```



```
[hadoop@master ~]$ hdfs dfsadmin -help
hdfs dfsadmin performs DFS administrative commands.
Note: Administrative commands can only be run with superuser permission.
The full syntax is:

hdfs dfsadmin
  [-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]
]
  [-safemode <enter | leave | get | wait>]
  [-saveNamespace [-beforeShutdown]]
  [-rollEdits]
  [-restoreFailedStorage true|false|check]
  [-refreshNodes]
  [-setQuota <quota> <dirname>...<dirname>]
  [-clrQuota <dirname>...<dirname>]
  [-setSpaceQuota <quota> [-storageType <storagetype>] <dirname>...<dirname>]
  [-clrSpaceQuota [-storageType <storagetype>] <dirname>...<dirname>]
  [-finalizeUpgrade]
  [-rollingUpgrade [<query|prepare|finalize>]]
  [-refreshServiceAcl]
  [-refreshUserToGroupsMappings]
  [-refreshSuperUserGroupsConfiguration]
  [-refreshCallQueue]
  [-refresh <host:ipc_port> <key> [arg1..argn]
  [-reconfig <namenode|datanode> <host:ipc_port> <start|status|properties>]
  [-printTopology]
  [-refreshNamenodes datanode_host:ipc_port]
  [-getVolumeReport datanode_host:ipc_port]
  [-deleteBlockPool datanode_host:ipc_port blockpoolId [force]]
  [-setBalancerBandwidth <bandwidth in bytes per second>]
  [-getBalancerBandwidth <datanode_host:ipc_port>]
  [-fetchImage <local directory>]
  [-allowSnapshot <snapshotDir>]
  [-disallowSnapshot <snapshotDir>]
  [-shutdownDatanode <datanode_host:ipc_port> [upgrade]]
  [-evictWriters <datanode_host:ipc_port>]
  [-getDatanodeInfo <datanode_host:ipc_port>]
```

```

[-metasave filename]
[-triggerBlockReport [-incremental] <datanode_host:ipc_port>]
[-listOpenFiles [-blockingDecommission] [-path <path>]]
[-help [cmd]]

-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]:
  Reports basic filesystem information and statistics.
  The dfs usage can be different from "du" usage, because it
  measures raw space used by replication, checksums, snapshots
  and etc. on all the DNs.
  Optional flags may be used to filter the list of displayed DNs.

-safemode <enter|leave|get|wait|forceExit>: Safe mode maintenance command.
  Safe mode is a Namenode state in which it
    1. does not accept changes to the name space (read-only)
    2. does not replicate or delete blocks.
  Safe mode is entered automatically at Namenode startup, and
  leaves safe mode automatically when the configured minimum
  percentage of blocks satisfies the minimum replication
  condition. Safe mode can also be entered manually, but then
  it can only be turned off manually as well.

-saveNamespace [-beforeShutdown]: Save current namespace into storage directories and
  reset edits
    log. Requires safe mode.
    If the "beforeShutdown" option is given, the NameNode does a
    checkpoint if and only if there is no checkpoint done during
    a time window (a configurable number of checkpoint periods).
    This is usually used before shutting down the NameNode to
    prevent potential fsimage/editlog corruption.

-rollEdits: Rolls the edit log.

-restoreFailedStorage: Set/Unset/Check flag to attempt restore of failed storage repli
cas if they become available.

-refreshNodes: Updates the namenode with the set of datanodes allowed to connect to th
e namenode.

  Namenode re-reads datanode hostnames from the file defined by
  dfs.hosts, dfs.hosts.exclude configuration parameters.
  Hosts defined in dfs.hosts are the datanodes that are part of

```

## dfsadmin 命令选项

命令选项	功能描述
-report	查看文件系统的基本信息和统计信息
-safeadmin enter leave get wait	安全模式命令。
-saveNamespace	可以强制创建检查点，仅仅在安全模式下面运行
-refreshNodes	重新读取 hosts 和 exclude 文件，使新的

	节点或需要退出集群的节点能够被 NameNode 重新识别。这个命令在新增节点或注销节点时用到。
-finalizeUpgrade	终结 HDFS 的升级操作。DataNode 删除前一个版本的工作目录，之后 NameNode 也这样做。
-upgradeProgress status   details   force	请求当前系统的升级状态   升级状态的细节   强制升级操作
-metasave filename	保存 NameNode 的主要数据结构到 hadoop.log.dir 属性指定的目录下的 <filename>文件中。
-setQuota <quota> <dirname> .....<dirname> :	为每个目录 <dirname> 设定配额 <quota>。目录配额是一个长整形整数，强制设定目录树下的 名字个数。
-clrQuota <dirname> .....<dirname>	为每个目录<dirname>清除配额设定
-fetchImage <lacal directory>	把最新的文件系统镜像文件从 元数据节点上下载到本地指定 目录
-clrQuota <dirname><dirname>	清除每个目录 dirname 的配额。以下情况会报错：(1) 这个目录不存在或者是一个文件 (2) 用户不是管理员
-restoreFailedStorage true false check	此选项将关闭自动尝试恢复故障的存储副本。如果故障的存储可用，再次尝试还原检

	查点 期间的日志编辑文件或文件系 统镜像 文件。“check” 选项将返 回当前设置
-help	查看帮助

**示例 1:** 显示文件系统的基本信息和统计信息

```
[hadoop@master ~]$ hdfs dfsadmin -report
```

```

[hadoop@master ~]$ hdfs dfsadmin -report
Configured Capacity: 81650475008 (76.04 GB)
Present Capacity: 42728058880 (39.79 GB)
DFS Remaining: 42728026112 (39.79 GB)
DFS Used: 32768 (32 KB)
DFS Used%: 0.00%
Replicated Blocks:
  Under replicated blocks: 3
  Blocks with corrupt replicas: 0
  Missing blocks: 0
  Missing blocks (with replication factor 1): 0
  Pending deletion blocks: 0
Erasure Coded Block Groups:
  Low redundancy block groups: 0
  Block groups with corrupt internal blocks: 0
  Missing block groups: 0
  Pending deletion blocks: 0

-----
Live datanodes (1):

Name: 172.17.0.2:9866 (0514918de0d6)
Hostname: 0514918de0d6
Decommission Status : Normal
Configured Capacity: 81650475008 (76.04 GB)
DFS Used: 32768 (32 KB)
Non DFS Used: 38922416128 (36.25 GB)
DFS Remaining: 42728026112 (39.79 GB)
DFS Used%: 0.00%
DFS Remaining%: 52.33%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Wed Sep 28 12:32:41 UTC 2022
Last Block Report: Wed Sep 28 04:23:49 UTC 2022
Num of Blocks: 3

[hadoop@master ~]$ █

```

示例 2: 获取安全模式

```
[hadoop@master ~]$ hdfs dfsadmin -safemode get
```

```
[hadoop@master ~]$ hdfs dfsadmin -safemode get
Safe mode is OFF
[hadoop@master ~]$
```

示例 3: 进入安全模式

[hadoop@master ~]\$ hdfs dfsadmin -safemode enter

```
[hadoop@master ~]$ hdfs dfsadmin -safemode enter
Safe mode is ON
[hadoop@master ~]$
```

示例 3: 可以强制创建检查点

[hadoop@master ~]\$ hdfs dfsadmin -saveNamespace

```
[hadoop@master ~]$ hdfs dfsadmin -saveNamespace
Save namespace successful
[hadoop@master ~]$
```

示例 4:

[hadoop@master ~]\$ hdfs dfsadmin -refreshNodes

```
Save namespace successful
[hadoop@master ~]$ hdfs dfsadmin -refreshNodes
Refresh nodes successful
[hadoop@master ~]$
```

● 1.4.1.14. Namenode 格式化升级回滚

运行 namenode 进行格式化、升级回滚等操作

namenode 选项和功能

命令选项	功能描述
-format	格式化元数据节点。先启动元数据节点，然后格式化，最后关闭

-upgrade	元数据节点版本更新后，应该以 upgrade 方式启动
-rollback	回滚到前一个版本。必须先停止集群，并且分发旧版本才可用
-importCheckpoint	从检查点目录加载镜像，目录由 fs.checkpoint.dir 指定
-finalize	持久化最近的升级，并把前一系统状态删除，这个时候再使用 rollback

#### ● 1.4.1.15. fsck 检查实用程序

fsck 命令运行 HDFS 文件系统检查实用程序，用于和 MapReduce 作业交互。

##### fsck 选项和功能

命令选项	功能描述
-path	检查这个目录中的文件是否完整
-move	移动找到的已损坏的文件到 /lost+found -rollback 回滚到前一个版本。必须先停止集群，并且分发旧版本才可用
-delete	删除已损坏的文件
-openforwrite	打印正在打开写操作的文件
-files	打印正在检查的文件名
-blocks	打印 block 报告（需要和 -files 参数一起使用）

-locations	打印每个 block 的位置信息 (需要和 -files 参数一起使用)
-racks	打印位置信息的网络拓扑图 (需要和 -files 参数一起使用)

### 示例 1: 检查目录中文件完整性

[hadoop@master ~]\$ hdfs fsck /test/dir0

```
[hadoop@master ~]$ hdfs fsck /test/dir0
Connecting to namenode via http://master:50070/fsck?ugi=hadoop&path=%2Ftest%2Fdir0
FSCK started by hadoop (auth:SIMPLE) from /172.17.0.2 for path /test/dir0 at Wed Sep 28
12:46:29 UTC 2022

/test/dir0/file2: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_107374182
7_1003. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s),
0 decommissioning replica(s).

Status: HEALTHY
Number of data-nodes: 1
Number of racks: 1
Total dirs: 1
Total symlinks: 0

Replicated Blocks:
Total size: 12 B
Total files: 1
Total blocks (validated): 1 (avg. block size 12 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 1 (100.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 2 (66.666664 %)

Erasure Coded Block Groups:
Total size: 0 B
Total files: 0
Total block groups (validated): 0
Minimally erasure-coded block groups: 0
Over-erasure-coded block groups: 0
Under-erasure-coded block groups: 0
Unsatisfactory placement block groups: 0
Average block group size: 0.0
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0
```

### 示例 2: 移动找到已损坏的文件



```
[hadoop@master ~]$ hdfs fsck -move
```

```
[hadoop@master ~]$ hdfs fsck /test/dir0
Connecting to namenode via http://master:50070/fsck?ugi=hadoop&path=%2Ftest%2Fdir0
FSCK started by hadoop (auth:SIMPLE) from /172.17.0.2 for path /test/dir0 at Wed Sep 28
12:46:29 UTC 2022

[hadoop@master ~]$ hdfs fsck -move
Connecting to namenode via http://master:50070/fsck?ugi=hadoop&move=1&path=%2F
FSCK started by hadoop (auth:SIMPLE) from /172.17.0.2 for path / at Wed Sep 28 12:47:27
UTC 2022

/test/dir0/file2: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_107374182
7_1003. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s),
0 decommissioning replica(s).

/test/file: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741825_1001
. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 deco
mmissioning replica(s).

/test/file2: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741826_100
2. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 dec
ommissioning replica(s).

Status: HEALTHY
Number of data-nodes: 1
Number of racks: 1
Total dirs: 4
Total symlinks: 0

Replicated Blocks:
Total size: 118 B
Total files: 8
Total blocks (validated): 3 (avg. block size 39 B)
Minimally replicated blocks: 3 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 3 (100.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 6 (66.666664 %)
```

**示例 3:** 删除已损坏的文件

```
[hadoop@master ~]$ hdfs fsck -delete
```

```

[hadoop@master ~]$ hdfs fsck -delete
Connecting to namenode via http://master:50070/fsck?ugi=hadoop&delete=1&path=%2F
FSCK started by hadoop (auth:SIMPLE) from /172.17.0.2 for path / at Wed Sep 28 12:48:08
UTC 2022

/test/dir0/file2: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_107374182
7_1003. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s),
0 decommissioning replica(s).

/test/file: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741825_1001
. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 deco
mmissioning replica(s).

/test/file2: Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741826_100
2. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 dec
ommissioning replica(s).

Status: HEALTHY
Number of data-nodes: 1
Number of racks: 1
Total dirs: 4
Total symlinks: 0

Replicated Blocks:
Total size: 118 B
Total files: 8
Total blocks (validated): 3 (avg. block size 39 B)
Minimally replicated blocks: 3 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 3 (100.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 6 (66.666664 %)

Erasure Coded Block Groups:
Total size: 0 B
Total files: 0
Total block groups (validated): 0

```

**示例 4:** 检查 HDFS 系统上/test 目录下的块信息和文件名

```

[hadoop@master ~]$ hdfs fsck /test -blocks -files

```

```
[hadoop@master ~]$ hdfs fsck /test -blocks -files
Connecting to namenode via http://master:50070/fsck?ugi=hadoop&blocks=1&files=1&path=%2Ftest
Ftest
FSCK started by hadoop (auth:SIMPLE) from /172.17.0.2 for path /test at Wed Sep 28 12:49:22 UTC 2022
/test <dir>
/test/dir0 <dir>
/test/dir0/file2 12 bytes, replicated: replication=3, 1 block(s): Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741827_1003. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
0. BP-863041297-172.17.0.2-1664331332122:blk_1073741827_1003 len=12 Live_repl=1

/test/dir1 <dir>
/test/dir1/file6 0 bytes, replicated: replication=3, 0 block(s): OK

/test/dir1/file7 0 bytes, replicated: replication=3, 0 block(s): OK

/test/file 94 bytes, replicated: replication=3, 1 block(s): Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741825_1001. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
0. BP-863041297-172.17.0.2-1664331332122:blk_1073741825_1001 len=94 Live_repl=1

/test/file1 0 bytes, replicated: replication=3, 0 block(s): OK

/test/file2 12 bytes, replicated: replication=3, 1 block(s): Under replicated BP-863041297-172.17.0.2-1664331332122:blk_1073741826_1002. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
```

- **1.4.1.16. pipe 运行管道作业**

pipe 命令运行管道作业

```
[hadoop@master ~]$ mapred pipes
```

```

[hadoop@master ~]$ mapred pipes
Usage: pipes
  [-input <path>] // Input directory
  [-output <path>] // Output directory
  [-jar <jar file>] // jar filename
  [-inputformat <class>] // InputFormat class
  [-map <class>] // Java Map class
  [-partitioner <class>] // Java Partitioner
  [-reduce <class>] // Java Reduce class
  [-writer <class>] // Java RecordWriter
  [-program <executable>] // executable URI
  [-reduces <num>] // number of reduces
  [-lazyOutput <true/false>] // createOutputLazily

Generic options supported are:
-conf <configuration file>      specify an application configuration file
-D <property=value>             define a value for a given property
-fs <file:///|hdfs://namenode:port> specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.
-jt <local|resourcemanager:port> specify a ResourceManager
-files <file1,...>              specify a comma-separated list of files to be copied
to the map reduce cluster
-libjars <jar1,...>             specify a comma-separated list of jar files to be included in the classpath
-archives <archive1,...>       specify a comma-separated list of archives to be unrar
chived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]

2022-09-28 13:02:49,408 INFO util.ExitUtil: Exiting with status 1: ExitException
[hadoop@master ~]$

```

## pipe 命令选项和功能

命令选项	功能描述
-conf<path>	配置作业
-jobconf <key=value>,<key=value> , ...	增加/覆盖作业的配置项
-input<path>	输入目录
-output<path>	输出目录
-jar<jar file>	<b>Jar 文件名</b>
-inputformat<class>	InputFormat 类
-map<class>	Java Map 类
-partitioner<class>	Java Partitioner

-reduce<class>	Java Reduce 类
-writer<class>	Java RecordWriter
-program<executable>	可执行程序 URI
-reduces<num>	分配 Reduce 任务的数量

#### ● 1.4.1.17. job 作业交互

该命令与 MapReduce 作业进行交互和命令。

[hadoop@master ~]\$ mapred job

```
hadoop@master ~]$ mapred job
Usage: job <command> <args>
  [-submit <job-file>]
  [-status <job-id>]
  [-counter <job-id> <group-name> <counter-name>]
  [-kill <job-id>]
  [-set-priority <job-id> <priority>]. Valid values for priorities are: VERY_HIGH HIGH NORMAL LOW VERY_LOW DEFAULT. In addition to this, integers also can be used.
  [-events <job-id> <from-event-#> <#-of-events>]
  [-history [all] <jobHistoryFile|jobId> [-outfile <file>] [-format <human|json>]]
  [-list [all]]
  [-list-active-trackers]
  [-list-blacklisted-trackers]
  [-list-attempt-ids <job-id> <task-type> <task-state>]. Valid values for <task-type> are MAP REDUCE. Valid values for <task-state> are pending, running, completed, failed, killed
  [-kill-task <task-attempt-id>]
  [-fail-task <task-attempt-id>]
  [-logs <job-id> <task-attempt-id>]
  [-config <job-id> <file>]

Generic options supported are:
  -conf <configuration file>          specify an application configuration file
  -D <property=value>                  define a value for a given property
  -fs <file:///|hdfs://namenode:port> specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.
  -jrt <local|resourcemanager:port>    specify a ResourceManager
  -files <file1,...>                  specify a comma-separated list of files to be copied to the map reduce cluster
  -libjars <jar1,...>                 specify a comma-separated list of jar files to be included in the classpath
  -archives <archive1,...>            specify a comma-separated list of archives to be unarchived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]

hadoop@master ~]$
```

## job 命令选项和功能

命令选项	功能描述
-submit<job-file>	提交作业
-status<job-id>	打印 map 和 reduce 完成百分比和所有 计数器
-counter<job-id><group-name><counter-name>	打印计数器的值
-kill<job-id>	杀死指定作业
-events <job-id> <from-event-#> <#-of-events>	打印给定范围内 jobtracker 接收到的事件细节
-history [all]	打印作业的细节，失败和被杀死原因和 细节提示。通过指定[all]选项，可以查看关于成功的任务和未完成的任务等 工作的更多细节。
-list [all]	-list all 显示所有作业。-list 只显示将要完成的作业。
-kill-task<task-id>	杀死任务。被杀死的任务不计失败的次数。
-fail-task<task-id>	使任务失败。被失败的任务不计失败的次数。
-logs<job-id><task-attempt-id>	更改作业的优先级。允许的优先级值是 VERY_HIGH、HIGH、

	NORMAL、LOW、和 VERY_LO
--	----------------------

## 1.4.2. Eclipse 插件开发配置

- 把我们的"hadoop-eclipse-plugin-2.7.0.jar"放到 Eclipse 的目录的"plugins"中，然 后重启 Eclipse 即可生效。

重启 Eclipse

上图中左侧"Project Explorer"下面发现"DFS Locations", 说明 Eclipse 已经识别刚才 放入的 Hadoop Eclipse 插件了。

- 选择"Window"菜单下的"Preference", 然后弹出一个窗体, 在窗体的左侧, 有 一列选项, 里面会多出"Hadoop Map/Reduce"选项, 点击此选项, 选择 Hadoop 在 Window 下的目录
- 切换 "Map/Reduce" 工作目录 选择 "Window"---" Perspective" ---"Open Perspective"--- "other" , 弹出一个窗体, 从 中选择"Map/Reduce"选项即可进行切换。

Location Name: Name 可以是任意, 标识一个"Map/Reduce Location" Map/Reduce (V2)

Master Host: 172.17.0.2 (Master.Hadoop 的 IP 地址)

Port: 9001

DFS Master Use M/R Master host: 前面的勾上。(因为我们的 NameNode 和

ResourceManager 都在一个机器上。)

Port: 9000 备注: 这里面 DFS Master 中的 Port 为在 “core-site.xml” 中 “fs.defaultFS” 配置 的端口, 其他可以保持默认值。

General Advanced parameters

Location name: HDFS

Map/Reduce(V2) Master

Host: 172.17.0.2

Port: 9001

DFS Master

☒ Use M/R Master host

Host: 172.17.0.2

Port: 9000

User name: 阿小橙

SOCKS proxy

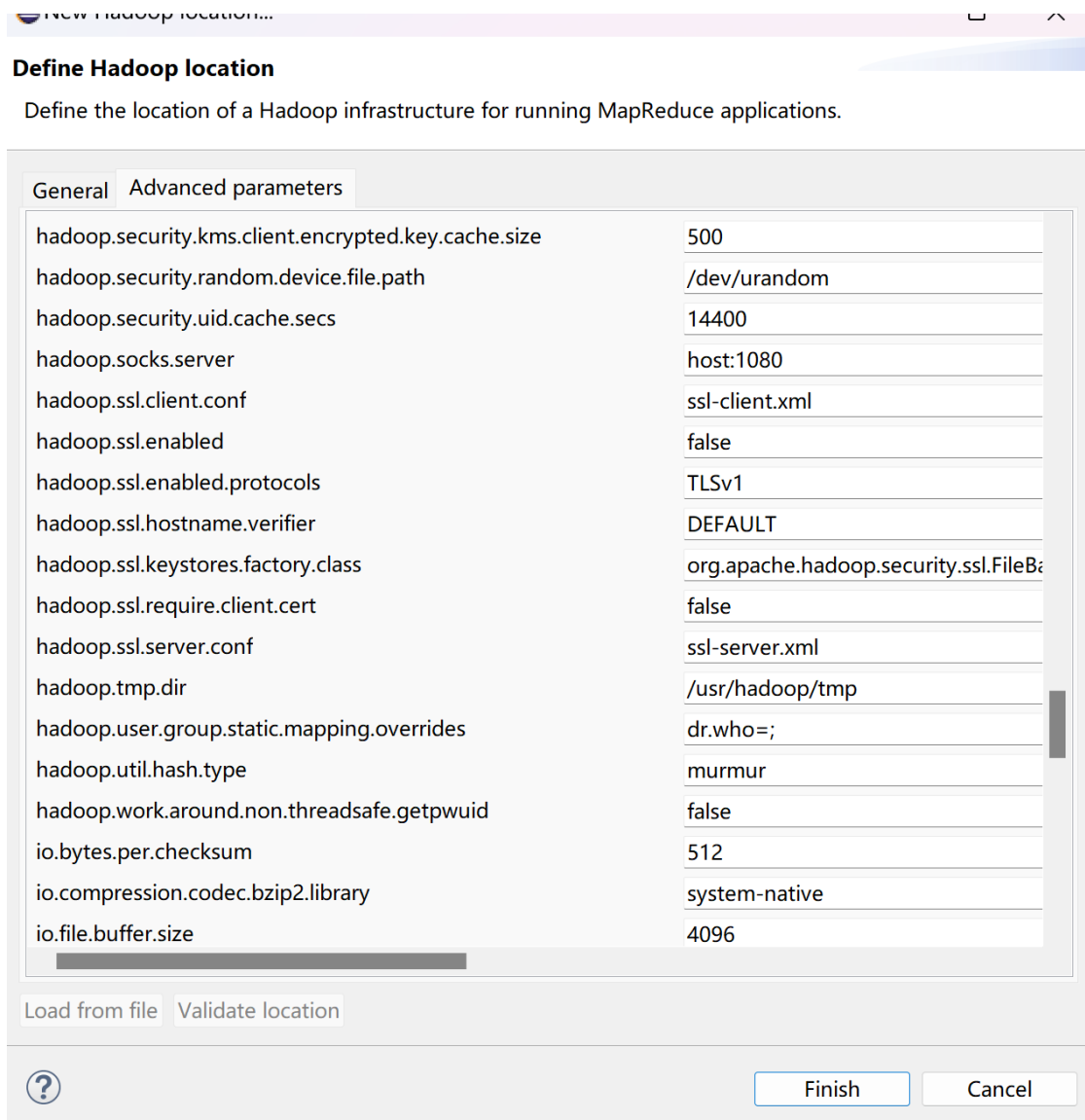
☐ Enable SOCKS proxy

Host: host

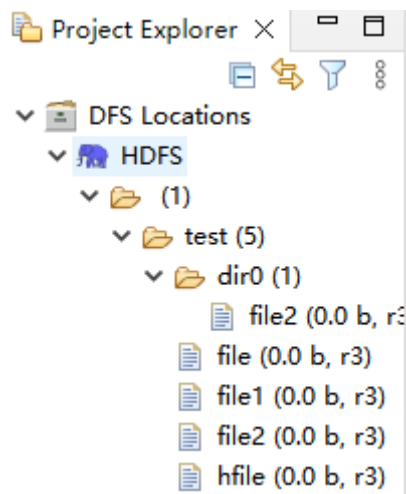
Port: 1080

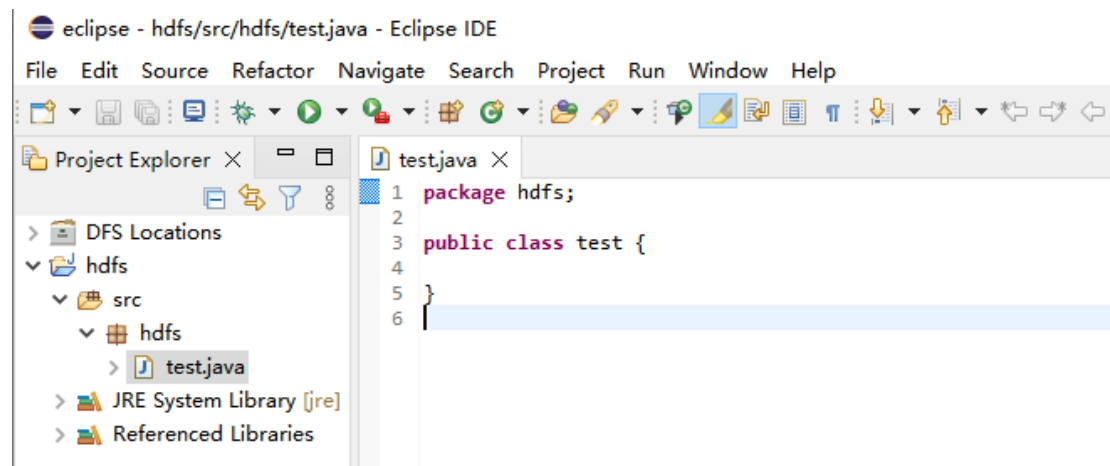
- 接着点击 "Advanced parameters" 从中找见 "hadoop.tmp.dir", 修改成为我们 Hadoop 集群中设置的目录, 我们的 Hadoop 集群是 "/usr/hadoop/tmp", 这个参数在 "core-site.xml" 中 "hadoop.tmp.dir" 进行了配置





- 查看 HDFS 文件系统，并尝试建立文件夹和上传文件。点击 Eclipse 软件左侧 的 "DFS Locations" 下面的 "HDFS"，就会展示出 HDFS 上的文件结构。





### 1.4.3. HDFS Java API 编程

通过连接伪分布式或完全分布式的 Master 节点，进行 HDFS Java API 的开发，连接的端口是 9000。

#### 1.4.3.1. 上传文件

通过 “FileSystem.copyFromLocalFile(Path src,Path dst)” 可将本地文件上传到 HDFS 指定的文件位置上.其中 src 和 dst 均为文件的完整路径

```
test.java × log4j.properties
1 package hdfs;
2 import java.io.IOException;
3 import java.net.URI;
4 import java.net.URISyntaxException;
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.fs.FileStatus;
7 import org.apache.hadoop.fs.FileSystem;
8 import org.apache.hadoop.fs.Path;
9 public class test {
10     public static void main(String[] args) throws Exception {
11         Configuration conf = new Configuration();
12         URI uri = new URI("hdfs://192.168.216.130:9000");
13         FileSystem fs = FileSystem.get(uri, conf);
14         //本地文件
15         Path src = new Path("D:\\file.txt");
16         //HDFS 存放位置
17         Path dst = new Path("/test/test.txt");
18         fs.copyFromLocalFile(src, dst);
19         System.out.println("Upload to " + conf.get("fs.defaultFS"));
20         //以下相当于执行 hdfs dfs -ls /
21         FileStatus files[] = fs.listStatus(dst);
22         for (FileStatus file : files) {
23             System.out.println(file.getPath());
24         }
25     }
26 }
```

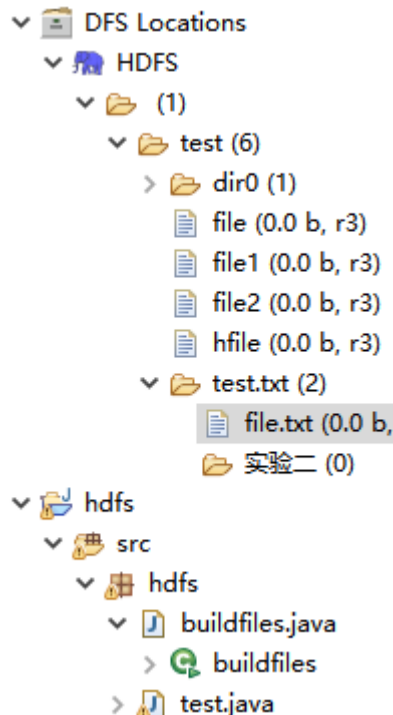
配置的 log4j.properties 文件

```
test.java log4j.properties ×
1 log4j.rootLogger=INFO, stdout
2 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
3 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
4 log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n
5 log4j.appender.logfile=org.apache.log4j.FileAppender
6 log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
7 log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n
```

修改文件权限以及关闭安全模式

```
[hadoop@master ~]$ hdfs dfsadmin -safemode leave
Safe mode is OFF
[hadoop@master ~]$ hdfs dfs -chmod 777 /test/
```

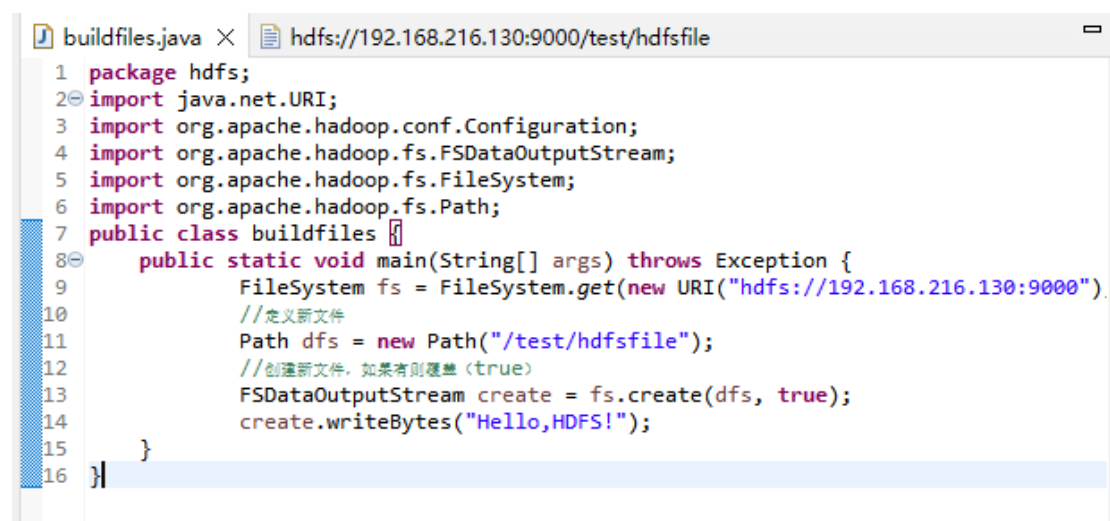
运行结果:



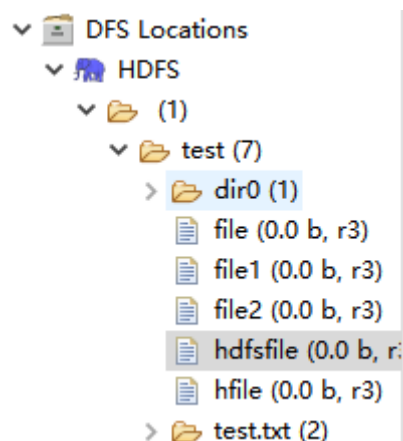
```
Upload to file:///
hdfs://192.168.216.130:9000/test/test.txt/实验二
hdfs://192.168.216.130:9000/test/test.txt/file.txt
```

### 1.4.3.2 创建文件:

通过 “`FileSystem.create(Path f, Boolean b)`” 可在 HDFS 上创建文件, 其中 `f` 为 文件的完整路径, `b` 为判断是否覆盖。

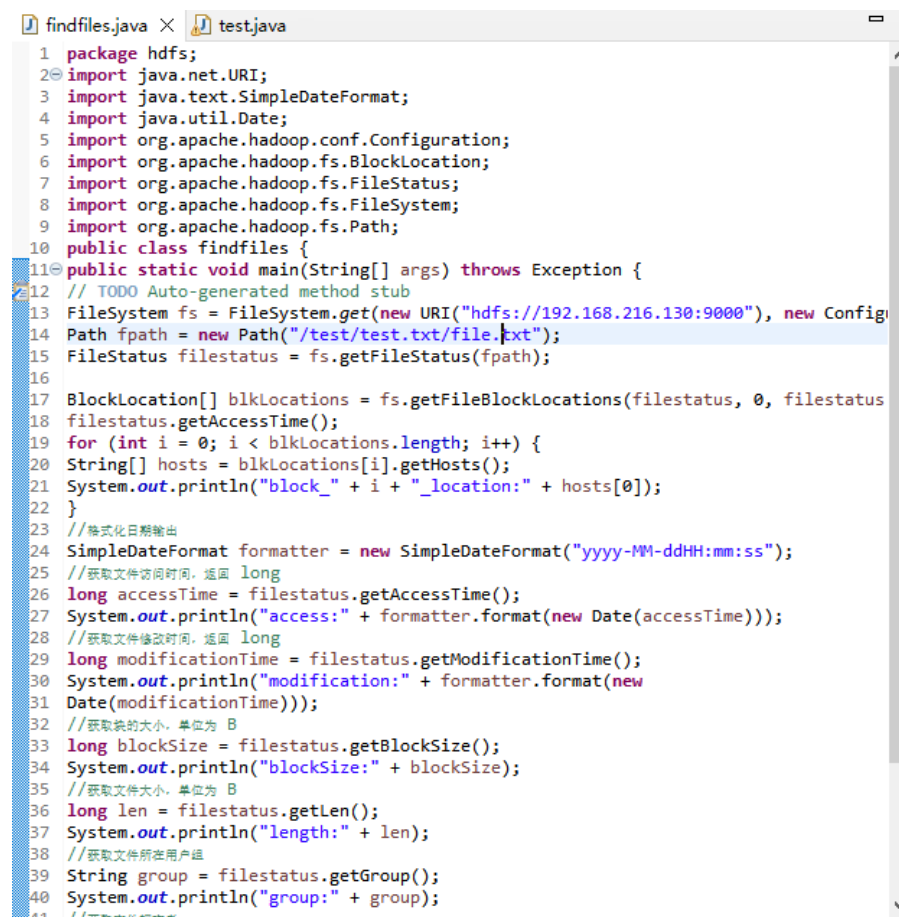


## 运行结果



### 1.4.3.3 查找文件信息：

通过 “Class FileStatus” 可查找指定文件在 HDFS 集群上的具体信息，包括访问时间、修改时间、文件长度、所占块大小、文件拥有者、文件用户组 and 文件复制数等信息。



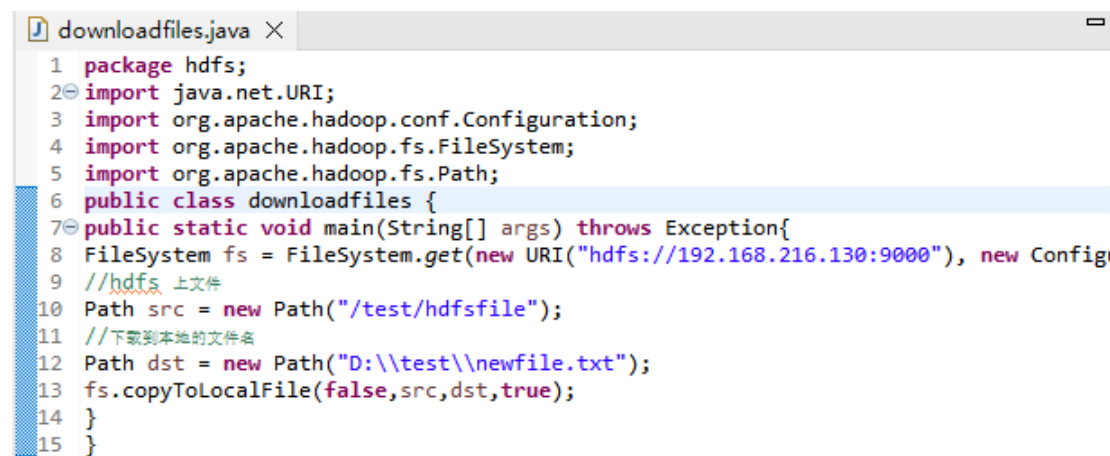
修改了 ip 地址以及查找的目标文件地址

## 实验结果

```
access:2022-10-14 15:27:38
modification:2022-10-14 15:27:38
blockSize:134217728
length:0
group:supergroup
owner:Administrator
replication:3
```

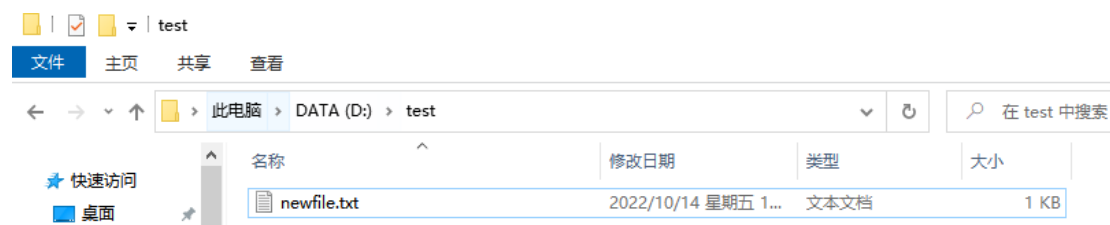
### 1.4.3.4 下载文件：

从 HDFS 下载文件到本地非常简单，直接调用 `FileSystem.copyToLocalFile(Path src, Path dst)` 即可。其中第一个参数为上传后是否删除原文件，第二个参数 `src` 为 HDFS 上的文件，`dst` 为要下载到本地的文件名，第四个参数为是否进行检验。



```
downloadfiles.java x
1 package hdfs;
2 import java.net.URI;
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.FileSystem;
5 import org.apache.hadoop.fs.Path;
6 public class downloadfiles {
7     public static void main(String[] args) throws Exception{
8         FileSystem fs = FileSystem.get(new URI("hdfs://192.168.216.130:9000"), new Config
9         //hdfs 上文件
10        Path src = new Path("/test/hdfsfile");
11        //下载到本地的文件名
12        Path dst = new Path("D:\\test\\newfile.txt");
13        fs.copyToLocalFile(false,src,dst,true);
14    }
15 }
```

## 实验结果



## 1.5 实验结果：

了解 Hadoop Shell 命令的用法。

熟悉 HDFS 中的 JAVA API 的使用。

通过实验掌握基本的 HDFS 中的 JAVA API 编程方法。

掌握 hadoop 分布式文件系统 Shell 命令的基本操作。

虽然过程遭遇了许多挫折，但还是坚持做了下来。

# 暨南大学本科实验报告专用纸(附页)

---