import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Scanner;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

/\*\*

\*

\* @author abhilasha : 17/05/2017

\* Session3.Assignment2

\*

\*/

public class MainClass {

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

{

//Instantiate configuration object and populate it with configuration files

Configuration conf = new Configuration();

conf.addResource(new Path("/usr/local/hadoop-2.6.0/etc/hadoop/core-site.xml"));

conf.addResource(new Path("/usr/local/hadoop-2.6.0/etc/hadoop/hdfs-site.xml"));

//Instantiate FileSystem object

FileSystem fs = FileSystem.get(conf);

Scanner scanner = new Scanner(System.in);

//\*\*\*\*\*\*\*\*\*\*\*\*Task4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Perform listing of directory

System.out.println("\*\*\*\*\*\*\*\*\*\*\* Task4 : Listing of directories \*\*\*\*\*\*\*\*\*\*\*\*");

performListingOfDirectory(fs,scanner);

//\*\*\*\*\*\*\*\*\*\*\*\* Task5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Display content of an hdfs file on screen

System.out.println("\*\*\*\*\*\*\*\*\*\*\* Task5 : Display the contents of a file in hdfs on screen \*\*\*\*\*\*\*\*\*\*\*\*");

readFile(fs,scanner);

//\*\*\*\*\*\*\*\*\*\*\*\* Task6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Copy a file from local system to hdfs

System.out.println("\*\*\*\*\*\*\*\*\*\*\* Task6 : Copy a file from local system to hdfs \*\*\*\*\*\*\*\*\*\*\*\*");

copyFile(fs,scanner);

fs.close();

scanner.close();

}

private static void performListingOfDirectory(FileSystem fs,Scanner scanner) throws Exception

{

//Initialise start and end time stamp

long lStartTime = 0;

long lEndTime = Long.MAX\_VALUE;

//Take directory, who listing is to be performed, as input

System.out.println("Enter the directory to perform list on : ");

String strPath = scanner.next();

//Instantiate the path variable using the input directory

Path path = new Path(strPath);

if(!fs.isDirectory(path))

{

System.out.println("Input path is not that of a directory");

throw new Exception("Provided path is that of a file when it was expecting that of a directory");

}

//Take time stamp range as input from user

System.out.println("Enter the last-modification timestamp range : ");

lStartTime = scanner.nextLong();

lEndTime = scanner.nextLong();

//Method to perform listing of directory

listFilesAndDirectories(fs,path,lStartTime,lEndTime);

}

private static void listFilesAndDirectories(FileSystem fs,Path path, long lStartTime, long lEndTime) throws FileNotFoundException, IOException

{

FileStatus[] aFileStatuses=fs.listStatus(path);

for(FileStatus filesStatus:aFileStatuses)

{

long lCurrentModificationTime = filesStatus.getModificationTime();

if(lCurrentModificationTime>= lStartTime && lCurrentModificationTime <= lEndTime )

{

System.out.println("Path : "+filesStatus.getPath());

System.out.println("Modification Time : "+lCurrentModificationTime);

System.out.println("Is Directory : "+filesStatus.isDirectory());

System.out.println("Length : "+filesStatus.getLen());

System.out.println("Permissions : "+filesStatus.getPermission().toString());

System.out.println();

}

if(filesStatus.isDirectory())

{

listFilesAndDirectories(fs, filesStatus.getPath(),lStartTime,lEndTime);

}

}

}

private static void readFile(FileSystem fs,Scanner scanner) throws Exception

{

//Take the file, which is to be read, as input

System.out.println("Enter the file to read : ");

String strPath = scanner.next();

//Instantiate the path variable using the input directory

Path path = new Path(strPath);

if(fs.isDirectory(path))

{

System.out.println("Input path is not that of a file");

throw new Exception("Provided path is that of a directory when it was expecting that of a file");

}

//Open the file to read

FSDataInputStream inputStream=fs.open(path);

System.out.println("File "+strPath+" opened to read");

System.out.println("Contents of the file "+strPath+" are as follows : ");

BufferedReader br = new BufferedReader(new InputStreamReader(inputStream));

String line =br.readLine();

while(line!=null)

{

System.out.println("Line Read = "+line);

line=br.readLine();

}

br.close();

inputStream.close();

}

private static void copyFile(FileSystem fs, Scanner scanner) throws IOException, Exception

{

//Take the file, which is to be copied, as input

System.out.println("Enter the file path from the local system [Source path]: ");

String strSrcPath = scanner.next();

//Instantiate the path variable using the input directory

Path srcPath = new Path(strSrcPath);

if(fs.isDirectory(srcPath))

{

System.out.println("Input path is not that of a file");

throw new Exception("Provided path is that of a directory when it was expecting that of a file");

}

//Take the destination directory as input

System.out.println("Enter the directory path from hdfs [Destination path]: ");

String strDestPath = scanner.next();

//Instantiate the path variable using the input directory

Path destPath = new Path(strDestPath);

if(!fs.isDirectory(destPath))

{

System.out.println("Input path is not that of a directory");

throw new Exception("Provided path is that of a file when it was expecting that of a directory");

}

System.out.println("list of the files/directories before copy is as follows : ");

listDirectoryContent(fs,destPath);

//Copy file from localsystem to hdfs

fs.copyFromLocalFile(false, srcPath, destPath);

System.out.println("Copy performed.\nlist of the files/directories before copy is as follows : ");

listDirectoryContent(fs,destPath);

}

/\*\*

\* This is a non-recursive listing

\* @param fs

\* @throws IOException

\* @throws FileNotFoundException

\*/

private static void listDirectoryContent(FileSystem fs,Path path) throws FileNotFoundException, IOException

{

FileStatus[] aFileStatuses=fs.listStatus(path);

System.out.println("Total Number of objects : "+aFileStatuses.length);

for(FileStatus filesStatus:aFileStatuses)

{

System.out.println(filesStatus.getPath());

}

}

}