文件的分割与合并

1. 利用**RandomAccessFile流实现文件的分割**:

思路：参数有：

1. **待分割的文件路径或File对象**，
2. **分割后子文件存放的文件夹路径或File对象**；
3. 分割的**单位大小**，也可以利用默认值1024。

具体过程：

1. 子文件数目计算：根据源文件的大小及分割单位计算出分割文件数目，
2. 将子文件的名字依次存放到List集合中；
3. 循环分割，子文件数目为循环次数。
4. 每一次分割需要参数有：第几次分割、指针位置、分割的大小（主要是最后一次分割大小不是分割单位）、目标文件夹。
5. 输入流采用RandomAccessFile流；输出流采用文件输出流（利用BufferedOutputStream装饰）。

注意：

a）最后一次分割文件大小不是分割单位大小；

b）读取数据的时候，最后一次读到缓冲数组中的数据较多，需要控制好输出的个数，确保每次写入的字节数目就是分割文件的大小。

**package** random.access.file;

**import** java.io.BufferedOutputStream;

**import** java.io.File;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.OutputStream;

**import** java.io.RandomAccessFile;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** SplitFile {

**private** **long** blockSize;

**private** **int** num;

**private** **long** fileSize;

**private** String filePath;

**private** File file;

**private** List<String> names = **new** ArrayList<String>();

**public** SplitFile(String filePath) {

**this**(filePath,1024);

}

**public** SplitFile(String filePath,**long** blockSize) {

**this**(**new** File(filePath),blockSize);

}

**public** SplitFile( File file,**long** blockSize) {

**this**.blockSize = blockSize;

**this**.file = file;

}

**public** **void** init(File destDir) {

**this**.initSize();

**this**.initPathName(destDir);

}

**public** **void** initSize() {

**if**(**this**.file == **null** ||**this**.file.isDirectory() || (!**this**.file.exists())) {

**return**;

}

**this**.fileSize = **this**.file.length();

**if** (**this**.fileSize < **this**.blockSize ) {

**this**.blockSize = **this**.fileSize;

}

**this**.num = (**int**) Math.*ceil*(**this**.fileSize\*1.0/**this**.blockSize);

}

**public** **void** initPathName(File destDir) {

String name = **this**.file.getName();

**for** (**int** i = 0; i < **this**.num; i++) {

**this**.names.add(name+".part"+(i+1));

}

}

**public** **void** split(String destDirPath) {

**this**.split(**new** File(destDirPath));

}

**public** **void** split(File destDir) {

init(destDir);

**for (int i = 0; i < this.num; i++) {**

**long begin = i\*this.blockSize;**

**long len = this.blockSize;**

**if (i == this.num -1) {**

**len = this.fileSize - this.blockSize\*i;**

**}**

**System.*out*.println(this.fileSize);**

**System.*out*.println(len);**

**this.splitDetail(destDir, i,begin,len);**

**}**

}

**public** **void** splitDetail(File destDir,**int** n,**long** begin,**long** len) {

File subFile = **new** File(destDir,**this**.names.get(n));

RandomAccessFile raf = **null**;

OutputStream os = **null**;

**try** {

raf = **new** RandomAccessFile(**this**.file, "rw");

os = **new** BufferedOutputStream(**new** FileOutputStream(subFile));

raf.seek(begin);

byte[] buff = new byte[10243];

int len1 = 0;

while(-1 !=(len1 = (raf.read(buff)))) {

if (len - len1 >= 0) {

os.write(buff,0,len1);

len -= len1;

}else {

os.write(buff, 0, (int) len);//关键

break;

}

os.flush();

}

} **catch** (IOException e) {

}**finally** {

**try** {

raf.close();

os.close();

} **catch** (IOException e) {

}

}

System.***out***.println("YYYYYYY"+subFile.length());

}

}

1. 原始的合并文件：

参数：（1） 子文件的文件夹或子文件的路径名字符串数组或子文件对象数组；

（2）合并文件的路径或File对象。

思路： （1） 首先获取子文件对象数组；

（2） 创建一个文件输出流（利用BufferedOutputStream装饰），并需要append；

（3）根据子文件的个数，循环。一次循环一个文件输入流，这里未引入序列流，下面会进行改善。读取的数据，都利用同一个输出流输出。

**package** random.access.file;

**import** java.io.BufferedInputStream;

**import** java.io.BufferedOutputStream;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**import** java.util.Arrays;

**public** **class** MergerFile {

**private** File[] files;

**private** String[] subFilePaths;

**private** String subFilesDirPathString;

**private** File subFilesDir;

**public** MergerFile(File subFilesDir) {

**this**.files = subFilesDir.listFiles();

System.***out***.println(Arrays.*toString*(files));

**this**.subFilesDir = subFilesDir;

}

**private** **int** subFileNum;

**private** String[] subFilesnames;

**private** File destFile;

**public** MergerFile(File[] files) {

**super**();

**this**.files = files;

}

**public** MergerFile(File[] files, File destFile) {

**super**();

**this**.files = files;

**this**.destFile = destFile;

}

**public** **void** merger(File destFile) {

OutputStream os = **null**;

**try** {

os = **new** BufferedOutputStream(**new** FileOutputStream(destFile,**true**));

**for**(File file:files) {

InputStream is = **null**;

**try** {

is = **new** BufferedInputStream(**new** FileInputStream(file));

**byte**[] buff = **new** **byte**[1024];

**int** len = 0;

**while**(-1 != (len = is.read(buff))) {

os.write(buff,0, len);

}

os.flush();

} **catch** (IOException e) {

}**finally**{

**try** {

is.close();

} **catch** (IOException e) {

}

}

}

} **catch** (FileNotFoundException e1) {

}**finally** {

**try** {

os.close();

} **catch** (IOException e) {

}

}

}

}

1. 利用**序列流**：

引入序列流后的改变：

**public** **void** merger(File destFile) {

OutputStream os = **null**;

InputStream is = **null**;

SequenceInputStream sis = **null**;

**try** {

os = **new** BufferedOutputStream(**new** FileOutputStream(destFile,**true**));

List<InputStream> list = **new** ArrayList<InputStream>();

for(File file:files) {

is = new BufferedInputStream(new FileInputStream(file));

list.add(is);

}

Enumeration<InputStream> e= Collections.*enumeration*(list);

sis = new SequenceInputStream(e);

**byte**[] buff = **new** **byte**[1024];

**int** len = 0;

**while**(-1 != (len = sis.read(buff))) {

os.write(buff,0, len);

}

os.flush();

} **catch** (Exception e1) {

}**finally** {

**try** {

sis.close();

is.close();

os.close();

} **catch** (IOException e) {

}

}

}

1. **多文件合并**利用**SequenceInputStream序列流**：
2. 示例1：利用Vector实现：

File file1 = new File("e:\\aaa\\1.txt");

File file2 = new File("e:\\aaa\\2.txt");

File file3 = new File("e:\\aaa\\3.txt");

File file4 = new File("e:\\aaa\\4.txt");

FileInputStream is1 = new FileInputStream(file1);

FileInputStream is2 = new FileInputStream(file2);

FileInputStream is3 = new FileInputStream(file3);

FileInputStream is4 = new FileInputStream(file4);

**Vector<FileInputStream> vec = new Vector<FileInputStream>();**

vec.add(is1);vec.add(is2);vec.add(is3);vec.add(is4);

**Enumeration<FileInputStream> en = vec.elements();**

SequenceInputStream sis = new SequenceInputStream(en);

FileOutputStream fos = new FileOutputStream("e:\\aaa\\zong.txt");

byte[] buf = new byte[1024\*1024];

int len = 0;

while((len = sis.read(buf))!= -1) {

fos.write(buf,0,len); fos.flush(); }

sis.close(); fos.close();

1. **利用ArrayList集合和集合工具类Collections中的enumeration方法。**

如： File file1 = new File("e:\\aaa\\1.txt");

File file2 = new File("e:\\aaa\\2.txt");

File file3 = new File("e:\\aaa\\3.txt");

File file4 = new File("e:\\aaa\\4.txt");

FileInputStream is1 = new FileInputStream(file1);

FileInputStream is2 = new FileInputStream(file2);

FileInputStream is3 = new FileInputStream(file3);

FileInputStream is4 = new FileInputStream(file4);

ArrayList<FileInputStream> al = new ArrayList<FileInputStream>();

al.add(is1);al.add(is2);al.add(is3);al.add(is4);

Enumeration<FileInputStream> en = Collections.enumeration(al);

**SequenceInputStream sis = new SequenceInputStream(en);**

FileOutputStream fos = new FileOutputStream("e:\\aaa\\zong2.txt");

byte[] buf = new byte[1024\*1024];

int len = 0;

while((len = sis.read(buf))!= -1) {

fos.write(buf,0,len);

fos.flush();

}

fos.close();

sis.close();

1. **文件切割和文件合并演示**：
2. **文件切割就是利用一个读取流，读取固定大小的数据然后顺序利用不同的输出流写数据到不同文件中。**

**也可以通过一个RandomAccessFile流，对文件进行随机读取。**

1. **文件合并就是利用序列流，把多个文件读取流合并起来，一个输出流，把数据都写到一个文件中去。**

代码：

public class SpiltFile {

private static final int SIZE = 1024\*1024;//1M

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

File dir = new File("e:\\bbb");

File file = new File(dir,"Joyce Chu - I MiSS U.mp4");

spiltFile(dir, file);

mergeFile(dir);

}

//文件合并

private static void mergeFile(File dir) throws IOException {

File[] files = dir.listFiles();

int num = files.length-1;

ArrayList<FileInputStream> al = new ArrayList<FileInputStream>();

for(int i = 1;i <= num;i++) {

FileInputStream fis = new FileInputStream(new File(dir,(i)+".part"));

al.add(fis);

}

Enumeration<FileInputStream> en = Collections.enumeration(al);

**SequenceInputStream sis = new SequenceInputStream(en);**

FileOutputStream fos = new FileOutputStream(new File("e:\\"+"好想你.mp4") );

byte[] buf = new byte[SIZE];

int len = 0;

while((len = sis.read(buf))!= -1) {

fos.write(buf,0,len);

fos.flush();

}

fos.close();

sis.close();

}

//文件切割

private static void spiltFile(File dir, File file)

throws Exception {

if(!file.exists())

file.createNewFile();

FileInputStream fis = new FileInputStream(file);

byte[] buf = new byte[SIZE];

int n = 0,i = 10, len = 0;

FileOutputStream fos = null;

while((len = fis.read(buf))!=-1) {

if(i <= 0)

i = 10;

if(i >= 10) {

**fos = new FileOutputStream(new File(dir,(++n)+".part"));**

}

fos.write(buf,0,len); i--;

}

fos.close(); fis.close();

}

}

1. 把文件切割和文件合并及**Properties集合**用于配置文件的综合练习：

**切割文件时，需要记录原文件的名称（包括格式），以及切割文件的个数，以便合并时回复到原来的文件。利用Properties文件记录。**

**代码：**

1. **文件名字过滤器：**

**public class SuffixFilter implements FilenameFilter {**

**String suffix ;**

**SuffixFilter(String suffix){**

**this.suffix = suffix;}**

**public boolean accept(File dir, String name) {**

**return name.endsWith(suffix); }}**

1. **文件切割器：**

**public class SpiltFileTest {**

**private static final int SIZE = 1024\*1024;**

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

**File dir = new File("e:\\ccc");**

**File file = new File(dir,"Joyce Chu - I MiSS U.mp4");**

**FileInputStream fis = new FileInputStream(file);**

**FileOutputStream fos = null;**

**byte[] buf = new byte[SIZE];**

**int len = 0, n = 10, count = 0;**

**while((len = fis.read(buf))!= -1) {**

**if(n <= 0) {**

**fos.close();**

**n = 10;**

**}**

**if(n == 10) {**

**fos = new FileOutputStream(new File(dir,(++count)+".part"));}**

**n--; fos.write(buf,0,len); fos.flush(); }**

**fis.close();**

**Properties prop = new Properties();**

**prop.setProperty("filename","Joyce Chu - I MiSS U.mp4");**

**prop.setProperty("spiltCount",Integer.toString(count));**

**File fileProp = new File(dir,"mani.properties");**

**fos = new FileOutputStream(fileProp);**

**prop.store(fos,"spiltInfo");**

**fos.close();**

**} }**

1. **文件合并器：**

**public class MergeFileTest {**

**private static final int SIZE = 1024\*1024;**

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

**Properties prop = new Properties();**

**File dir = new File("e:\\ccc");**

**File[] files = dir.listFiles(new SuffixFilter(".properties"));**

**if( files.length != 1 )**

**throw new RuntimeException("配置文件不存在或者文件个数不对。");**

**FileInputStream fis = new FileInputStream(files[0]); prop.load(fis);**

**String filename = prop.getProperty( "filename" );**

**int count = Integer.parseInt( prop.getProperty( "spiltCount" ));**

**fis.close();**

**ArrayList<FileInputStream> al = new ArrayList<FileInputStream>();**

**files = dir.listFiles(new SuffixFilter(".part"));**

**if(count != files.length)**

**throw new RuntimeException("碎片文件数目不对。");**

**for (int i = 0; i < count; i++ ) {**

**fis = new FileInputStream(files[i]);**

**al.add(fis);**

**}**

**Enumeration<FileInputStream> en = Collections.enumeration(al);**

**SequenceInputStream sis = new SequenceInputStream(en);**

**byte[] buf = new byte[SIZE];**

**FileOutputStream fos = new FileOutputStream(new File(dir,"copy"+filename));**

**int len = 0;**

**while((len = sis.read(buf)) != -1) {**

**fos.write(buf,0,len); fos.flush();**

**}**

**sis.close(); fos.close();**

**}**

**}**