# 十二章、输入输出流

把节点流包入缓冲流

## 1.File类

### (1)构造方法

路径名：文件 路径名：字符串

获取文件本身的一些信息，不涉及对文件的读写操作。文件目录和文件。

创建一个File对象的构造方法有3个：

File(String filename);

File(String path,String filename);

path 路径：绝对路径："D:\\io\\helloworld.txt"

相对路径："helloworld.txt"

File(File f, String filename);

[File](mk:@MSITStore:C:\Users\Administrator.LS--20161013LKV\Desktop\参考API\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#File(java.net.URI))([URI](mk:@MSITStore:C:\Users\Administrator.LS--20161013LKV\Desktop\参考API\JDK_API_1_6_zh_CN.CHM::/java/net/URI.html) uri) 相对路径：

### (2) 文件的属性

1>访问文件名：

public String getName() 获取文件名

public String getPath() 获取文件路径 路径名字符串

public String getAbsolutePath() 获取文件的绝对路径。 路径名字符串

public file [getAbsoluteFile](mk:@MSITStore:C:\Users\Administrator.LS--20161013LKV\Desktop\参考API\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#getAbsoluteFile())() 抽象路径名—>绝对路径名形式。

public String getParent() 获取文件的上一级目录。

boolean [renameTo](mk:@MSITStore:C:\Users\Administrator.LS--20161013LKV\Desktop\参考API\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#renameTo(java.io.File))([File](mk:@MSITStore:C:\Users\Administrator.LS--20161013LKV\Desktop\参考API\JDK_API_1_6_zh_CN.CHM::/java/io/File.html) dest) 重命名为file2，并转移file1

file1.renameTo(file2) file2为空, 预备放file1

2>文件检测

public boolean exits() 文件是否存在。

public boolean canRead() 文件是否是可读的。

public boolean canWrite() 文件是否可被写入。

public boolean isFile() 文件是否是一个普通文件，而不是目录。

public boolean isDirectroy() 文件是否是一个目录。

3>获取常规信息

public long length() 获取文件的长度（单位是字节）。

public long lastModified() 获取文件最后修改的时间。

### (3)文件目录的属性（包括上面）

1>读取目录下的全部文件的文件

public String[] list()

public File [] listFiles()

2>读取目录下指定类型的文件

public String[] list(FilenameFilter obj)

public File [] listFiles(FilenameFilter obj)

上述两方法的参数FilenameFilter是一个接口，该接口有一个方法：

public boolean accept(File dir,String name);

### (4)文件的创建与删除

File file1 = **new** File("D:\\io\\helloworld.txt");

1>删除当前文件

file1.delete();

2>创建文件

**if**(!file1.exists()){

**boolean** b = file1.createNewFile();

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

}

3>创建文件目录

public boolean mkdir() 创建本级目录

public boolean mkdir() 创建多级目录

## 2.io基本概念

### (1) Java IO原理

Java程序中，对于数据的输入/输出操作以”流(stream)” 的方式进行。

**输入input：**读取外部数据（磁盘、光盘等存储设备的数据）到程序（内存）中。

**输出output：**将程序（内存）数据输出到磁盘、光盘等存储设备中

### (2) 流的分类

1>按操作数据单位不同分为：

**字节流(8 bit)/二进制流** ：所有

**字符流(16 bit)** ：处理文本

2>按数据流的**流向**不同分为：**输入流，输出流**



IO流共涉及40多个类，都是从如下4个抽象基类派生的。

3>按流的角色的不同分为：

**节点流**：

直接作用文件的 **文件流**（最基础：所有都可以包含）

FileInputStream FileOutputStream

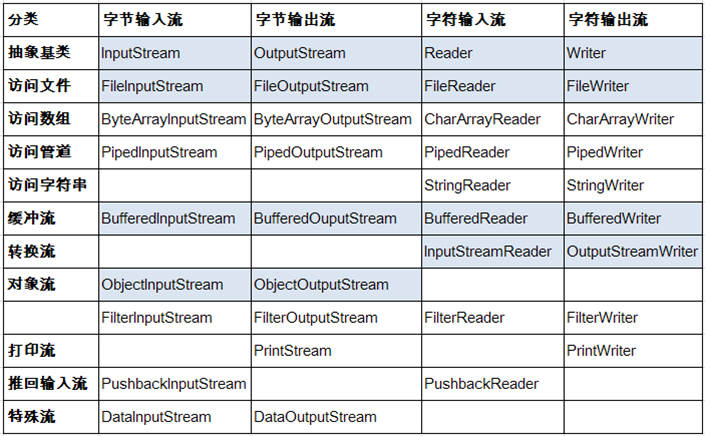
FileReader FileWriter

**数组流，管道流，字符串流**

**处理流**：

处理流是“连接”在已存在的流（节点流或处理流）之上，通过对数据的处理为程序提供更为强大的读写功能。





### (3)节点流一般方法

1>创建一个文件：不管什么操作，需要首先创建一个文件预放置

2>确定流，读或写：流； read(),readLine() ；write() ;

readObject() ；writeObject(p)

3>关闭： close() flush(),close();

4>实现文件的复制

### (4)带进度条的输入流

输入流类ProgressMonitorInputStream创建的输入流在读取文件时会弹出一个显示读取速度的进度条。它的构造方法是：

ProgressMonitorInputStream(Conmponent c,String s,InputStream);

## 3.文件流

### (1) 文件字节输入流 非文本文件

1>创建

构造方法： FileInputStream(String name);

FileInputStream(File file); **byte**[] b = **new** **byte**[5];

2>读文件read()

int read(); 读取一个字符

int read(byte b[]); 读取一个字符数组

int read(byte b[],int off,int len);

字符数组中起始于偏移量off处取出的len个字符写到文件

3>关闭流

close（）方法

Eg：FileInputStream fis = **null**;

**try** {

File file = **new** File("hello.txt");

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

**byte**[] b = **new** **byte**[5];// 读取到的数据要写入的数组。

**int** len; // 每次读入到byte中的字节的长度

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

String str = **new** String(b, 0, len);

System.***out***.print(str);

}

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**if** (fis != **null**) {

**try** {

fis.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

### (2)文件字节输出流

1>创建

构造方法：FileOutputStream（String name）

FileOutputStream(File file)

2>以字节为单位写文件，write方法

void write(int n) 写入一个字符。

void write(char b[]) 写入一个字符数组

void write(char b[],int off,int length)

字符数组中起始于偏移量off处取出的len个字符写到文件

3>关闭流

flush(),close（）方法

eg: File file = **new** File("hello2.txt");

FileOutputStream fos = **null**;

**try** {

fos = **new** FileOutputStream(file); 字符串转字节

fos.write(**new** String("\n"+"I love java 000").getBytes());

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**if** (fos != **null**) {

**try** {

fos.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

### (3) 文件字节输入流 文本文件

FileReader **char**[] b = **new** **char**[20];

### (4) 文件字节输入流

FileWriter

## 4.缓冲流 --专门用于文件的复制

### (1)BufferedInputStream和BufferedOutputStream类

把节点流包入缓冲流

BufferedInputStream bis = **new** BufferedInputStream(fis);

BufferedOutputStream bos = **new** BufferedOutputStream(fos);

//具体实现文本复制

**byte**[] b = **new** **byte**[20]; **char**[] b = **new** **char**[20];

**int** len ;

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

fos.write(b, 0, len );

}

### (2)BufferedReader和BufferedWriter类

读写文件的方法

readLine() BufferedReader流读取文本行

write(String s,int off,int len) BufferedWriter流把字符串s写到文件中

newLine(); BufferedWriter流向文件写入一个回行符

//具体实现文本复制

String str = **null**;

**while** ((str = br.readLine()) != **null**) {

bw.write(str);

bw.newLine(); //换行

bw.flush();

}

## 5.转换流 ——字节流🡪字符流 处理字符编码

### (1) 字符编码

解码：字节数组🡪字符串 InputStreamReader

编码：字符串🡪字节数组 OutputStreamWriter

### (2)一般处理

InputStreamReader isr = **new** InputStreamReader(fis,"UTF-8");

OutputStreamWriter osw = **new** OutputStreamWriter(fos,"UTF-8");

## 6.对象流

### (1)序列化与反序列化

**序列化(Serialize)：**用ObjectOutputStream类将一个Java对象写入IO流中

要求：1.类以及类的属性实现Serializable接口或Externalizable接口——序列化对象

2.指定静态变量private static final long serialVersionUID;

3.不能序列化static和transient修饰的成员变量

**反序列化(Deserialize)：**用ObjectInputStream类从IO流中恢复该Java对象

### (2) 对象克隆

一个对象调用clone()方法就可以获取该对象的克隆对象。

对象输入流通过对象的序列化信息来得到当前对象的一个克隆。

### (3)使用流

相关方法：类似缓冲流

writeObject(Object obj) 将一个对象obj写入到一个文件

ObjectOutputStream oos = **new** ObjectOutputStream(fos);

oos.writeObject(p);

readObject() 读取一个对象到程序中

ObjectInputStream ois = **new** ObjectInputStream(fis);

Person p = (Person)ois.readObject();

## 7. 随机流

使用RandomAccessFile类来创建一个随机访问文件流。其创建的流的指向既可以作为源也可以作为目的地。其构造方法：

RandomAccessFile(String name,String mode) ; r,rw,rwd,rws

RandomAccessFile(File file,String mode) ;

相关方法：

seek(long a) 定位RandomAccessFile流的读写位置

getFilePointer() 获取流的当前读写位置

P306 RandomAccessFile类的常用方法

## 8.标准输入输出流System

System.in和System.out 控制台（默认输入设备是键盘，输出设备是显示器）

System.setIn()和System.setOut() 任何地方

## 9.文件类的操作

### (1) 运行可执行文件

1>用Runtime 类声明一个对象( Runtime类在java.lang包)

Runtime ec;

2>然后使用该类的getRuntime()静态方法创建这个对象：

ec=Runtime.getRuntime();

3>ec可以调用exec(String command)方法打开本地机的可执行文件或执行一个操作。

### (2)文件锁

FileLock、FileChannel类处理Java提供的文件锁功能。

输入、输出流读写文件时可以使用文件锁。

RondomAccessFile创建的流在读写文件时使用文件锁的步骤如下：

(1)先使用RondomAccessFile流建立指向文件的流对象，该对象的读写属性必须是rw，例如：

RandomAccessFile input=new RandomAccessFile("Example.java","rw");

(2)Input流调用方法getChannel()获得一个连接到地层文件的FileChannel对象（信道），例如

FileChannel channel=input.getChannel();

(3)信道调用tryLock()或lock()方法获得一个FileLock（文件锁）对象，这一过程也称作对文件加锁，例如：FileLock lock=channel.tryLock();

### (3)文件使用Scanner解析

1>使用默认分隔标记解析文件

创建Scanner对象，并指向要解析的文件，例如：

File file = new File("hello.java");

Scanner sc = new Scanner(file); sc将空白作为分隔标记

相关方法：

next() 依次返回file中的单词

hasNext() 判断file最后一个单词是否已被next()方法返回

2>使用正则表达式作为分隔标记解析文件

创建Scanner对象，指向要解析的文件，并使用useDelimiter方法指定正则表达式作为分隔标记，例如:

File file = new File("hello.java");

Scanner sc = new Scanner(file);

sc.useDelimiter(正则表达式); sc将正则表达式作为分隔标记

相关方法：

next() 依次返回file中的单词

hasNext() 判断file最后一个单词是否已被next()方法返回

参看API方法类似：

数据流toByteArray()、数组流

打印流、随机流