什么是IO流

什么是异常

IO流体系

基本概念

异常体系

IO流操作异常

怎么处理异常禅常

使用步骤

抛出

IO流的使用

字符流

异常

文件

举例

捕获

字节流

学习章节

1. 异常概述
2. 异常处理方式
3. IO流
4. File类
5. 字符流读写文件
6. 字节流读写文件

异常概述

什么是异常

非正常情况，程序出现的错误

异常的分类

1. 异常（Exception）合理的应用程序可能需要捕获的问题

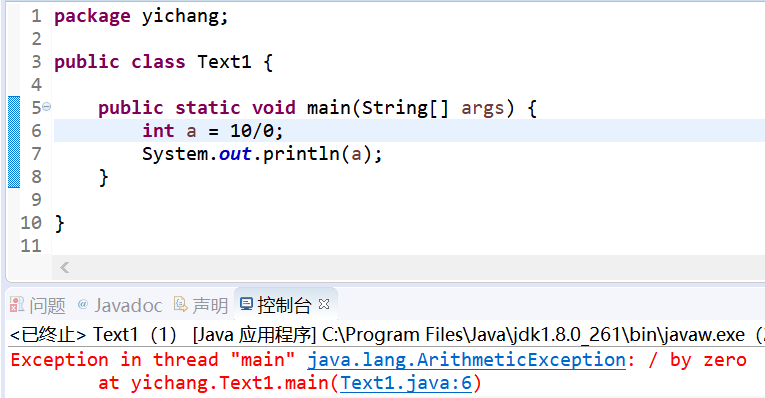
举例：NullPointerException，代码能处理

1. 错误（Error）合理的应用程序不应该试图捕获的问题

举例：StackOverFlowError，内存问题报错没法代码处理

异常处理的方式

1. JVM默认的处理方式：在控制台打印错误信息，并终止程序



1. 开发过程中异常的处理方式：

Try{

尝试执行的代码

}catch（Exception）{

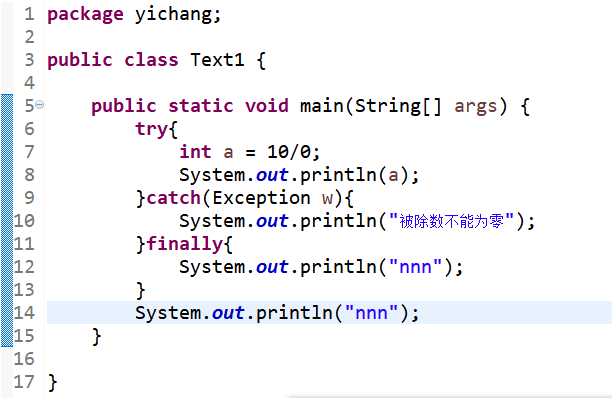
出现可能的异常之后的处理代码

}（finally）{

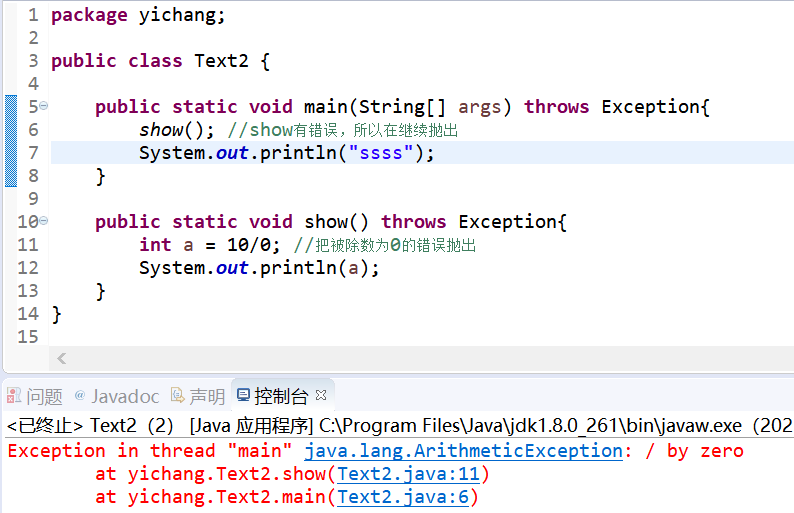
正常情况下一定会执行的代码，如关闭资源

}  
处理完异常之后，程序后面也会执行

捕获，自己处理



throws：抛出，交给调用者处理，且后面代码不在执行



IO流

定义：输入（input）输出（output），指的是数据像液体一样进行流动

作用：在本地磁盘和网络上操作数据

分类：按数据流向：输入流——读数据

输出流——写数据

按操作方式：字节流——以字节为单位来操作数据

InputStream 字节输入流的顶层抽象类

FileInputStream 普通的字节流输入

BufferedInputStream 高效的字节流输入

（字节缓冲输入流）

OutputStream 字节输出流的顶层抽象类

FileOutputStream 普通的字节流输出

BufferedOutputStream 高效的字节流输出

（字节缓冲输出流）

字符流——以字符为单位来操作数据

Reader 字符输入流的顶层抽象类

FileInputReader 普通的字节流输入

BufferedIReader 高效的字节流输入

（字节缓冲输入流）

Writer 字符输出流的顶层抽象类

FileInputWriter 普通的字节流输入

BufferedIWriter 高效的字节流输入

（字节缓冲输入流）

File类

定义：文件、文件夹，一个File都代表磁盘上的某个文件夹

构造方法：File（String pathname）

File（String father ，String child）

File（File father ，String child）

成员方法：createNewFile（）：创建文件

mkdir（）和mkdirs（）：创建目录 后者为多级

isDirectory（）：判断File对象是否为目录（文件夹）

isFile（）：判断File对象是否为文件

exists（）：判断File对象是否存在

不存在就建立，然后返回true，存在就返回false，属于判断

getAbsolutePath（）：获取绝对路径，以盘符开头的路径

例如：D：/1.txt

getPath（）：获取相对路径，一般是相对于当前项目路径来讲的

例如：1.txt

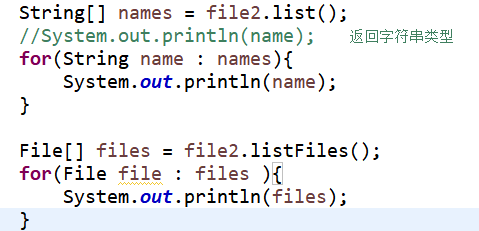
getName（）：获取文件名

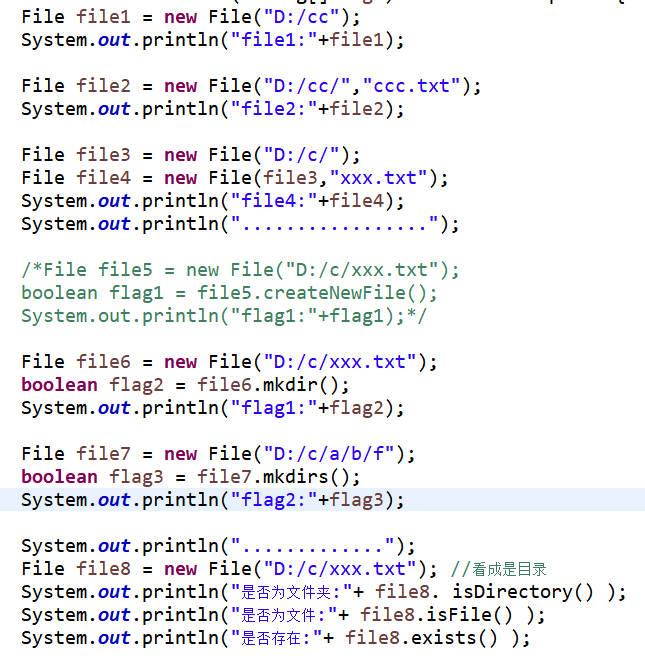
list（）：获取指定目录下所有文件（夹）名称数组

名称数组String[]

listFile（）：获取指定目录下所有文件（夹）File数组

File对象数组File[]





文件的创建与删除

Public booleam delete（）；

如:File.delete()

运行可执行文件

首先使用Runtime类声明一个对象，再用该类的getRuntime（）静态方法创建这个对象

如：Runtime 2

2 = Runtime.getRuntime（）

字符流读写文件

一、按单个字符读取

方式：1、创建字符流读文件对象

Reader reader = new FileReader（“数据源文件路径”）

Reader抽象类，new具体的子类 FileReader

2、调用方法读取数据

Int data = reader.read（）；

读取一个字符，返回该字符有效字符数，如a返回97.

若到达底部，返回-1

3、异常处理

throws IOException

4、关闭资源

Reader.close()

/\*ch = reader.read()) ！= -1

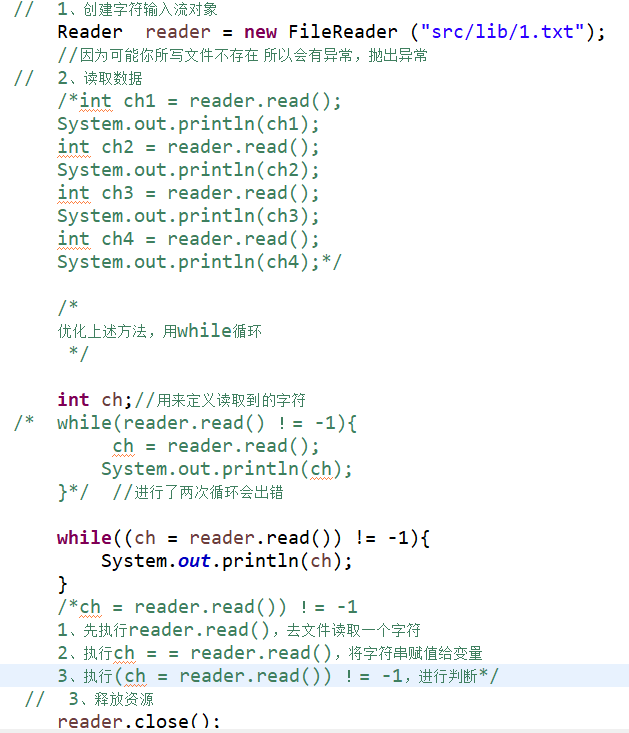
1、先执行reader.read()，去文件读取一个字符

2、执行ch = = reader.read()，将字符串赋值给变量

3、执行(ch = reader.read()) ！= -1，进行判断\*/

总结：创建字符输入流对象——定义读取到的字符——while循环

——关闭资源



二、按字符数组读取

方式：1、创建字符流读文件对象

Reader reader = new FileReader（“数据源文件路径”）

Reader抽象类，new具体的子类 FileReader

2、调用方法读取数据

char chs [] = new char[2048]

Int data = reader.read（chs）；

读取一个字符，返回该字符代表整数，如a返回97.

若到达底部，返回-1

3、异常处理

throws IOException

4、关闭资源

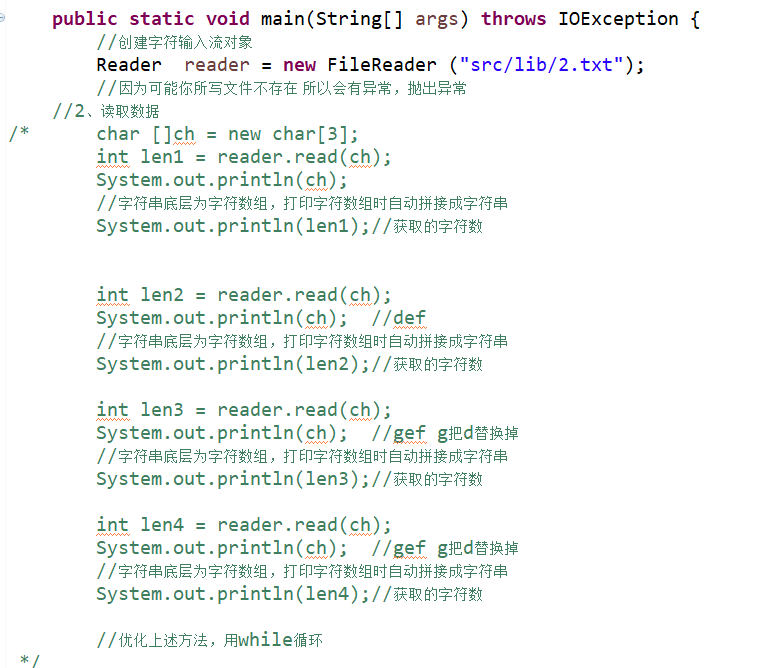
Reader.close()

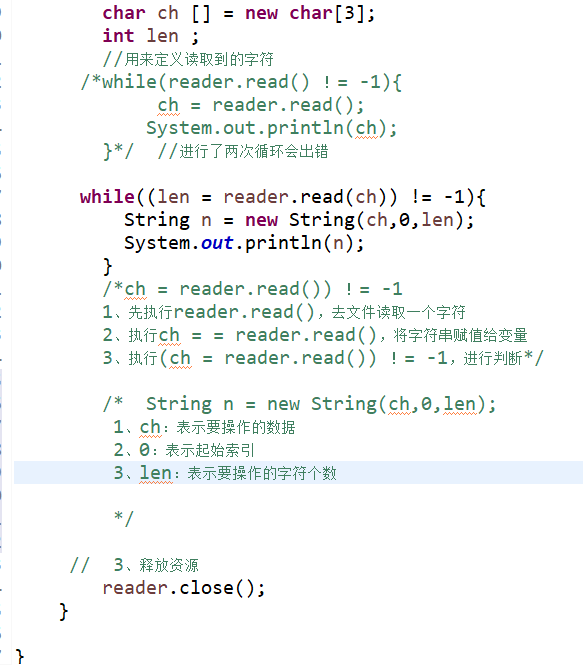
String n = new String(ch,0,len);

1、ch：表示要操作的数据

2、0：表示起始索引

3、len：表示要操作的字符个数





1. 按字符数组写入

方式：1、创建字符流写文件对象

Writer writer = new FileWriter（“数据源文件路径”）

Writer抽象类，new具体的子类 FileWriter

2、调用方法读取数据

char chs [] = {“陈”,”哈”,”哈”}

writer .write(chs) 写一个字符

3、异常处理

throws IOException

4、关闭资源

Reader.close()

1. 按字符串写入

方式：1、创建字符流写文件对象

Writer writer = new FileWriter（“数据源文件路径”）

Writer抽象类，new具体的子类 FileWriter

2、调用方法写数据

writer .write(“陈哈哈”) 写一个字符串

3、异常处理

throws IOException

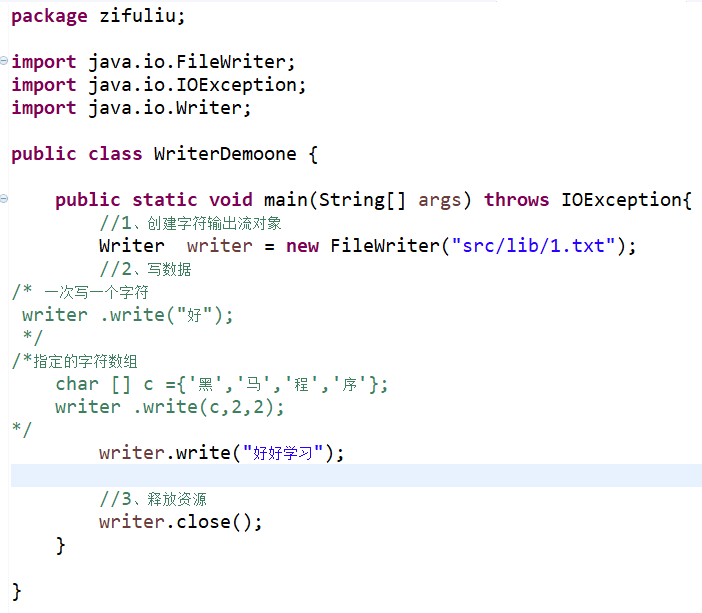
4、关闭资源

Reader.close()

void write(int n)一次写一个字符

void write(byte b[], int off, int len)一次写一个特定的字符数组

void wirte(byte b[])一次写一个字符串



字符流拷贝文件（综合）

如1.txt拷贝到2.txt，就要用reader读取1.txt的内容，通过writer写进2.txt

一、按单个字符读写

如1.txt拷贝到2.txt，就要用reader读取1.txt的内容，通过writer写进2.txt

方式：1、创建字符流读文件对象

Reader reader = new FileReader（“数据源文件路径”）

2、创建字符流写文件对象

Writer writer = new FileWriter（“数据源文件路径”）

3、调用方法读取数据

Int data = reader.read（）；

4、调用方法写数据

writer .write(int) 写一个字符串

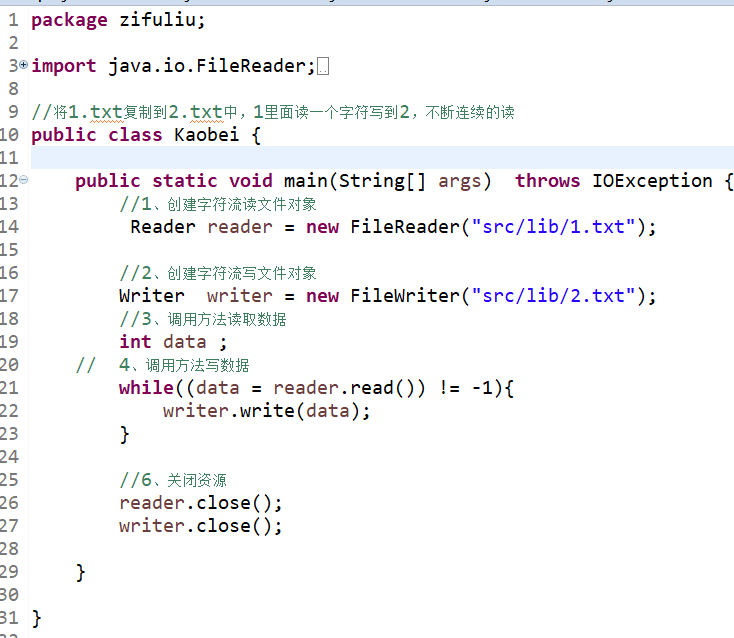
5、异常处理

throws IOException

6、关闭资源

reader.close()

writer.close()



二、按字符数组读写

方式：1、创建字符流读文件对象

Reader reader = new FileReader（“数据源文件路径”）

2、创建字符流写文件对象

Writer writer = new FileWriter（“数据源文件路径”）

3、 创建数组调用方法读取数据

char chs [] = new char[2048]

Int data = reader.read（chs）；

4、调用方法写数据

writer .write(chs,0,len)

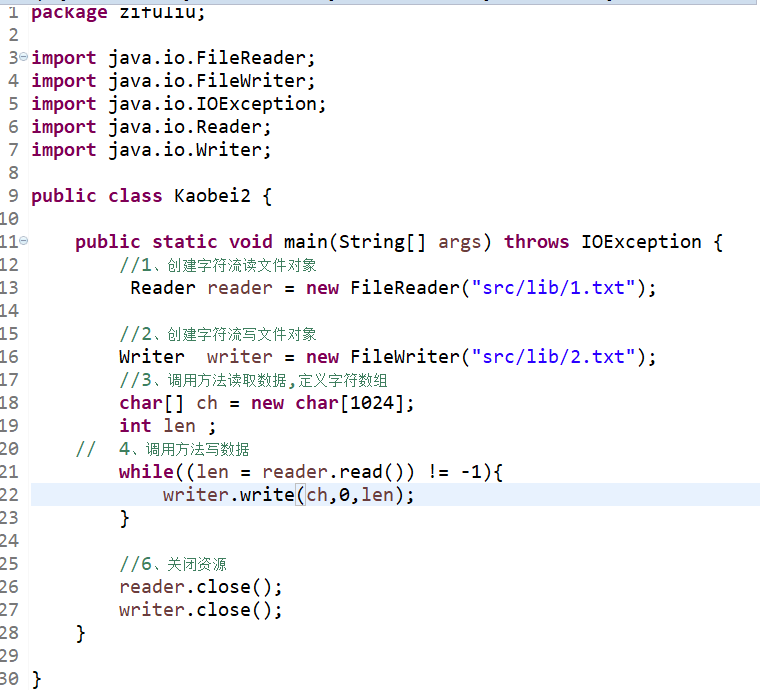
5、异常处理

throws IOException

6、关闭资源

reader.close()

writer.close()



字符缓冲流拷贝文件

（即使没有手动定义字符数组，底层也可以按字符数组读写）读取纯文本

BufferedReader（Reader reader）

Public string readLine（）一次读取一行数据并返回读取到的内容，否则返回null

BufferedWriter（ Writer writer）

Public void newLine（）根据当前相应操作系统给出换行符

方式：1、创建字符流读文件对象

BufferReader br = new BufferedReader（ new FileReader（“数据源文件路径”））

2、创建字符流写文件对象

BufferWriter b = new BufferedWriter（ new FileWriter（“数据源文件路径”））

3、定义变量

Int len；

4、使用while循环读写数据

While（（len = br.read（） ！= -1）{

b.write（len）；

}

5、异常处理

throws IOException

6、关闭资源

Br.close()

B.close()

字节流读写文件

1. 按单个字节读写

FileInputStream(String name);

FIleInputStream(File file);

FileOutputStream(String name);

FileOutputStream(File file);

方式：1、创建字符流读文件对象

InputStream br = new FileInputStream （“数据源文件路径”）

2、创建字符流写文件对象

OutputStream b = new FileOutputStream（“数据源文件路径”）

3、使用while循环读写数据

Int v；

While（（len = br.read（） ！= -1）{

b.write（len）；

}

4、异常处理

throws IOException

5、关闭资源

Br.close()

B.close()



1. 按字节数组读写

方式：1、创建字符流读文件对象

InputStream br = new FileInputStream （“数据源文件路径”）

2、创建字符流写文件对象

OutputStream b = new FileOutputStream（“数据源文件路径”）

3、定义字节数组

Byte b [] = new byte()

4、使用while循环读写数据

Int v；

While（（len = br.read（） ！= -1）{

b.write（b，0，len）；

}

5、关闭资源

Br.close()

B.close()

字节缓冲流拷贝文件

方式：1、创建字符流读文件对象

BufferInputStream br = new BufferInputStream （ new FileReader（“数据源文件路径”））

2、创建字符流写文件对象

BufferOutputStream b = new BufferOutputStream（ new FileWriter（“数据源文件路径”））

3、定义变量

Int len；

4、使用while循环读写数据

While（（len = br.read（） ！= -1）{

b.write（len）；

}

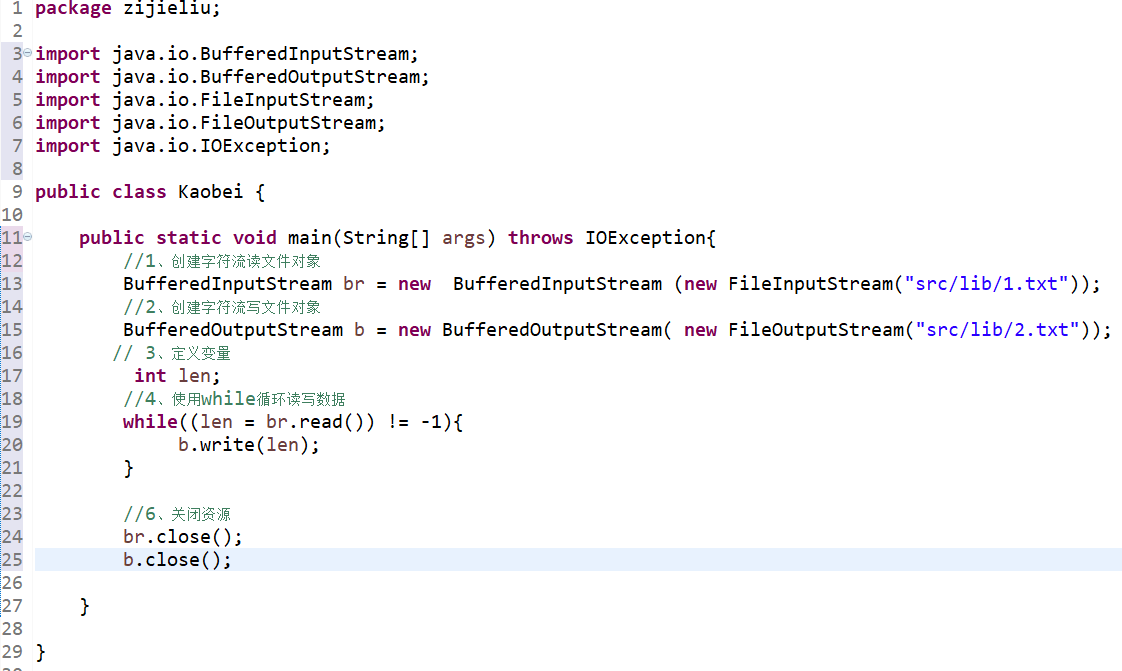
5、异常处理

throws IOException

6、关闭资源

Br.close()

B.close()



总结：拷贝纯文本字符流，其他的(图片/视频/音频）字节流