

# 第二十一章 IO数据访问

## 1. I/O流概述

- 1. I/O（Input/Output）：读写
- 2. 流：从源节点到目标节点数据的流动
- 3. 源节点：称为输入流，用来读取数据
- 4. 目标节点：称为输出流，用来写入数据
- 5. 举例：百度网盘下载资料到本地
  - a. 源节点：百度网盘
  - b. 目标节点：本地电脑

## 2. 流的分类

- 1. 从传输的角度来说，流分为字节流和字符流
- 2. InputStream和OutputStream是字节流的父类
- 3. Reader和Writer字符流的父类

分类	字节流	字符流
输出流	InputStream	Reader
输入流	OutputStream	Writer
流中的数据	二进制	字符 字符串

## 3. 使用字符流读取文件

### 3.1 相对路径和绝对路径

- *FileNotFoundException*
- 相对路径和绝对路径

```

1  package com.powernode.reader01;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileReader;
5
6  public class Test01 {
7      public static void main(String[] args) {
8          /**
9           * 1. FileNotFoundException: 文件没找到
10          *    1. 文件不存在
11          *    2. 路径错误
12          * 2. 绝对路径和相对路径
13          *    1. 绝对路径: 从盘符开始到文件, 看到路径就知道具体位置
14          *       D:\powernode\02-JavaSE\03-
code\JavaProject\day21\src\fileTest.txt
15          *    2. 相对路径: 从当前模块到文件, 看到路径不知道具体位置
16          *       .\day21\src\fileTest.txt
17          *       也可以写成 (省略.\)
18          *       day21\src\fileTest.txt
19          */
20          try {
21              FileReader fileReader = new FileReader("day21/src/fileTest.txt");
22          } catch (FileNotFoundException e) {
23              throw new RuntimeException(e);
24          }
25      }
26  }

```

## 3.2 字符流逐个读取文件中数据

代码块

```

1  package com.powernode.reader01;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileReader;
5  import java.io.IOException;
6
7  public class Test02 {
8      public static void main(String[] args) {
9          try {
10             //1. 创建文件读取对象, 并指定需要读取文件的路径
11             FileReader fileReader = new FileReader("./day21/src/fileTest.txt");
12             //2. 读取文件, 返回一个int类型的数据
13             System.out.println(fileReader.read()); //97 是 a 的ASCII
14             System.out.println(fileReader.read()); //98 是 b 的ASCII

```

```

15         System.out.println(fileReader.read()); //99 是 c 的ASCII
16         System.out.println(fileReader.read()); // -1 ,没有读取到数据
17     } catch (FileNotFoundException e) {
18         System.out.println("文件没有找到");
19     } catch (IOException e) {
20         throw new RuntimeException(e);
21     }
22 }
23 }

```

### 3.3 循环读取

代码块

```

1  package com.powernode.reader01;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileReader;
5  import java.io.IOException;
6
7  public class Test03 {
8      public static void main(String[] args) {
9          FileReader fileReader = null;
10         try {
11             //1.创建一个文件读取对象，并指定需要读取的文件路径（建立了读取通道）
12             fileReader = new FileReader("./day21/src1/fileTest.txt");
13             //2.读取文件
14             int read = fileReader.read();
15             while(read != -1){
16                 System.out.println((char) read);
17                 read = fileReader.read();
18             }
19         } catch (FileNotFoundException e) {
20             throw new RuntimeException(e);
21         } catch (IOException e) {
22             throw new RuntimeException(e);
23         } finally {
24             //3.关闭流
25             try {
26                 if (fileReader != null)
27                     fileReader.close();
28             } catch (IOException e) {
29                 throw new RuntimeException(e);
30             }
31         }
32     }

```

### 3.4 使用字符数组做缓存进行优化

//3. 创建字符数组做缓存

```
char[] cbuf = new char[2];
```

//4. 读取数据到缓存，返回读取的字符个数，读取完毕返回-1

```
int read = fileReader.read(cbuf);
```

```
while(read != -1){
```

```
    for (int i = 0; i < cbuf.length; i++) {
```

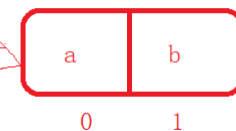
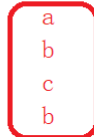
```
        char c = cbuf[i];
```

```
        System.out.println(c);
```

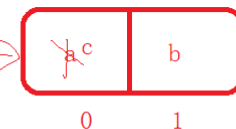
```
    }
```

```
    read = fileReader.read(cbuf);
```

```
}
```



分开写，其实就一个数组



#### 代码块

```
1 package com.powernode.reader01;
2
3 import java.io.FileNotFoundException;
4 import java.io.FileReader;
5 import java.io.IOException;
6
7 public class Test04 {
8     public static void main(String[] args) {
9         //1. 确定文件读取路径
10        String path = "./day21/src/fileTest.txt";
11        FileReader fileReader = null;
12        try {
13            //2. 创建文件读取对象
14            fileReader = new FileReader(path);
15            //3. 创建字符数组做缓存
16            char[] cbuf = new char[2];
17            //4. 读取数据到缓存，返回读取的字符个数，读取完毕返回-1
18            int read = fileReader.read(cbuf);
19            while(read != -1){
20                for (int i = 0; i < read; i++) { //读取几个输出几个
21                    char c = cbuf[i];
22                    System.out.println(c);
23                }
24                read = fileReader.read(cbuf);
25            }
26        } catch (IOException e) {
27            e.printStackTrace();
28        }
29    }
30 }
```

```

26         } catch (FileNotFoundException e) {
27             throw new RuntimeException(e);
28         } catch (IOException e) {
29             throw new RuntimeException(e);
30         }finally {
31             if (fileReader != null) {
32                 try {
33                     fileReader.close();
34                 } catch (IOException e) {
35                     throw new RuntimeException(e);
36                 }
37             }
38         }
39     }
40 }

```

## 4. 使用字符流写入文件

### 4.1 把字符串写入文件

代码块

```

1  package com.powernode.writer02;
2
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6
7  public class Test01 {
8      public static void main(String[] args) {
9          //1.确定文件写入路径
10         String path = "./day21/src/fileTest.txt";
11         FileWriter fileWriter = null;
12         try {
13             //2.创建文件写入对象，并建立写入通道
14             fileWriter = new FileWriter(path);
15             /**
16              * 3.写入文件:
17              *     1.清空文件
18              *     2.写入缓存 (通道)
19              */
20             fileWriter.write("hello world");
21         } catch (IOException e) {
22             throw new RuntimeException(e);
23         }finally {

```

```

24         //4.关闭流
25         //4.1清空缓存-物理写入
26         //4.2关闭写入通道
27         if (fileWriter != null) {
28             try {
29                 fileWriter.close();
30             } catch (IOException e) {
31                 throw new RuntimeException(e);
32             }
33         }
34     }
35 }
36 }

```

## 4.2 使用flush实现物理写入

- 什么时候使用flush？什么时候使用close？

代码块

```

1  package com.powernode.writer02;
2
3  import java.io.FileWriter;
4  import java.io.IOException;
5
6  public class Test02 {
7      public static void main(String[] args) {
8          //1.确定文件写入路径
9          String path = "./day21/src/fileTest.txt";
10         FileWriter fileWriter = null;
11         try {
12             //2.创建文件写入对象，并建立写入通道
13             fileWriter = new FileWriter(path);
14             /**
15              * 3.写入文件：
16              *     1.清空文件
17              *     2.写入缓存（通道）
18              */
19             fileWriter.write("下班去哪里玩? ");
20             //刷新缓存，写入硬盘
21             fileWriter.flush();
22             fileWriter.write("去后海吧! ");
23             fileWriter.flush();
24             /**
25              * 若需要【立即写入】数据且继续使用流，必须调用flush()。
26              * 若操作结束要关闭流，close()会自动处理 flush，无需额外调用。

```

```

27         * 对于带缓冲的流，仅靠close()虽然能保证最终数据写入，但无法满足实时性要求
    (如网络通信、实时日志)。
28         */
29     } catch (IOException e) {
30         throw new RuntimeException(e);
31     }
32 }
33 }

```

## 4.3 文件续写

- `true`表示追加内容到文件末尾处（续写）

代码块

```

1  package com.powernode.writer02;
2
3  import java.io.FileWriter;
4  import java.io.IOException;
5
6  public class Test03 {
7      public static void main(String[] args) {
8          //1.确定文件写入路径
9          String path = "./day21/src/fileTest.txt";
10         FileWriter fileWriter = null;
11         try {
12             //2.创建文件写入对象
13             fileWriter = new FileWriter(path,true); //true表示追加内容到文件末尾处
    (续写)
14             //3.写入文件
15             fileWriter.write("\n下班去哪里玩? ");
16         } catch (IOException e) {
17             throw new RuntimeException(e);
18         } finally {
19             if (fileWriter!=null) {
20                 try {
21                     fileWriter.close();
22                 } catch (IOException e) {
23                     throw new RuntimeException(e);
24                 }
25             }
26         }
27     }
28 }

```

## 5. 字符流实现文件备份

代码块

```
1  package com.powernode.backup03;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileReader;
5  import java.io.FileWriter;
6  import java.io.IOException;
7
8  public class Test01 {
9      public static void main(String[] args) {
10         /**
11          * - 文件备份, copy一个副本
12          *     1. 读取文件
13          *     2. 创建目标文件
14          *     3. 写入目标文件
15          */
16         //1. 确定读写文件路径
17         String sourcePath = "./day21/src/com/powernode/backup03/Test01.java";
18         String targetPath = sourcePath + ".bak";
19         FileReader fileReader = null;
20         FileWriter fileWriter = null;
21         try {
22             //2. 创建文件读写对象
23             fileReader = new FileReader(sourcePath);
24             fileWriter = new FileWriter(targetPath);
25             //3. 创建一个字符串数组缓存
26             char[] cbuf = new char[500];
27             int read = fileReader.read(cbuf);
28             while(read != -1){
29                 //5. 写入文件
30                 fileWriter.write(cbuf, 0, read);
31                 read = fileReader.read(cbuf);
32             }
33         } catch (FileNotFoundException e) {
34             throw new RuntimeException(e);
35         } catch (IOException e) {
36             throw new RuntimeException(e);
37         } finally {
38             //6. 关闭流 (先创建后关闭)
39             if (fileWriter != null){
40                 try {
41                     fileWriter.close();
42                 } catch (IOException e) {
```



```

43         throw new RuntimeException(e);
44     }
45 }
46 if (fileReader != null) {
47     try {
48         fileReader.close();
49     } catch (IOException e) {
50         throw new RuntimeException(e);
51     }
52 }
53
54 }
55 }
56 }

```

## 6. 使用try()对代码进行优化

代码块

```

1  package com.powernode.backup03;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileReader;
5  import java.io.FileWriter;
6  import java.io.IOException;
7  class Student implements AutoCloseable{
8      @Override
9      public void close() throws Exception {
10         System.out.println("Student.close");
11     }
12 }
13 public class Test02 {
14     public static void main(String[] args) {
15         /**
16          * - 文件备份, copy一个副本
17          *     1. 读取文件
18          *     2. 创建目标文件
19          *     3. 写入目标文件
20          */
21         //1. 确定读写文件路径
22         String sourcePath = "./day21/src/com/powernode/backup03/Test01.java";
23         String targetPath = sourcePath + ".bak";
24         try (
25             //1.try()创建文件读写对象, try执行完毕最后会自动调用close方法关闭资源
26             FileReader fileReader = new FileReader(sourcePath);
27             FileWriter fileWriter = new FileWriter(targetPath);

```

```

28         /**
29         * try() 中创建的对象, 该类必须实现AutoCloseable
30         * 重新AutoCloseable的close() 方法会自动执行
31         */
32         Student student = new Student();
33     ){
34         //创建字符数组作为缓存
35         char[] cbuf = new char[2];
36         //读取文件
37         int read = fileReader.read(cbuf);
38         while(read != -1){
39             //写入文件
40             fileWriter.write(cbuf,0,read);
41             read = fileReader.read(cbuf);
42         }
43     } catch (FileNotFoundException e) {
44         throw new RuntimeException(e);
45     } catch (Exception e) {
46         throw new RuntimeException(e);
47     }
48
49
50 }
51 }

```

## 7. 使用字节流copy视频

代码块

```

1  package com.powernode.backup03;
2
3  import java.io.*;
4
5  public class Test03 {
6      public static void main(String[] args) {
7          //1.确定文件读写路径
8          String sourcePath = "D:\\powernode\\02-JavaSE\\04-video\\day01\\01-认识
Java语言.mp4";
9          String targetPath = "D:\\01-认识Java语言.mp4";
10
11
12         try(
13             //2.创建文件读写对象
14             FileInputStream fileInputStream = new
FileInputStream(sourcePath);

```

```
15         FileOutputStream fileOutputStream = new
FileOutputStream(targetPath);
16     ) {
17         //3.提供缓存
18         byte[] bbuf = new byte[1024];
19         //4.读取文件
20         int read = fileInputStream.read(bbuf);
21         while(read!=-1){
22             //5.写入文件
23             fileOutputStream.write(bbuf,0,read);
24             read = fileInputStream.read(bbuf);
25         }
26     } catch (FileNotFoundException e) {
27         throw new RuntimeException(e);
28     } catch (IOException e) {
29         throw new RuntimeException(e);
30     }
31 }
32 }
```

8.