河南中医药大学信息技术学院（智能医疗行业学院）产教协同课程建设成果

智能医学工程专业《医疗信息系统开发》课程

第02章：文件读写

黄子杰

河南方和信息科技有限公司

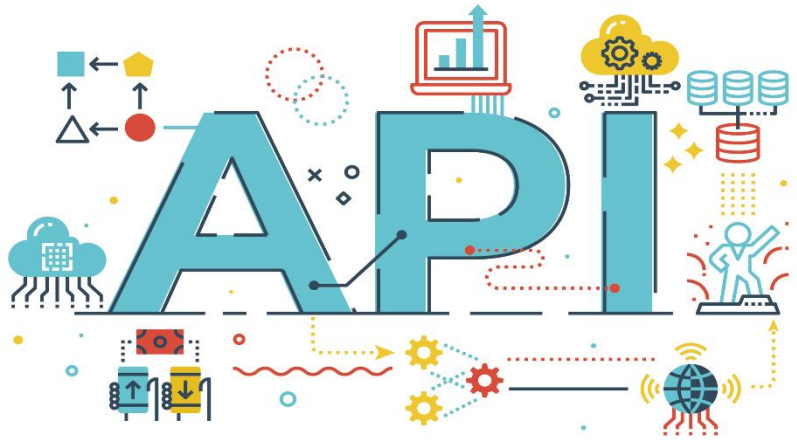
河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

https://aitcm.hactcm.edu.cn

2025/9/18

本章概要

- 什么是文件
  - 非易失性存储
  - 数据的集合
  - 文件的元数据
  - 操作系统的抽象
  - 文件的分类
  - 流
- 为什么要读写文件
  - 持久化存储
- **怎样读写文件**
  - java.io.File
  - 文本文件的读写（字符流）
  - 二进制文件的读写（字节流）



什么是文件

- 从一个普通用户的视角看，文件是电脑桌面上那些带着图标的东西，比如“论文.docx”、“成绩单.xlsx”、“照片.jpg”。我们可以双击打开它们，编辑、保存、删除。

- 从技术角度，我们需要一个更精确、更深入的定义。

- **核心定义**：文件是操作系统提供的，一种在**非易失性存储介质**（如硬盘、固态硬盘SSD、U盘）上**组织、存储和管理数据**的机制。

什么是文件

非易失性存储

- 对比内存（RAM）：程序运行时，数据在内存中进行计算。内存是易失性的，一旦断电，里面的所有数据都会丢失。

- 文件的作用：文件存储在硬盘等介质上，是非易失性的。即使断电，文件内容也会永久保存。这就是“持久化（Persistence）”的含义。文件是程序和数据跨越时间存在的载体。

什么是文件

数据的集合

- 文件本质上是一个字节序列。它就是一连串的0和1。

- 文件可以存储任何类型的数据。这些数据如何被解读，取决于创建和使用它的程序。

  同一串字节，文本编辑器会试图把它解读成文字。

  图片查看器会把它解读成像素的颜色信息。

  音乐播放器会把它解读成声波的振动信息。

- 文件的扩展名（如 .txt, .jpg）只是一个给用户的提示，它约定俗成地告诉用户和操作系统“这个文件大概率是什么类型”。但操作系统并不会阻止我们将图片数据重命名为 .txt 来打开，当然，打开后看到的会是乱码。

讨论: 什么是数据, 什么是元数据?

什么是文件

- 文件不仅仅是数据本身，还包含了一系列描述文件本身的属性信息，称为元数据。这就像一本书，不仅有内容（数据），还有书名、作者、页数（元数据）。
- 常见的文件元数据包括：

  文件名：文件的标识符。

  文件类型：通常由扩展名指示（.java, .class, .exe）。

  大小：文件占用的存储空间，单位是字节（Byte）。

  路径：文件在文件系统中的确切位置（如 D:\test\report.doc）。

  创建时间、最后修改时间、最后访问时间。

  权限：哪些用户可以读、写或执行这个文件。

  隐藏/可见属性。

在Java中，java.io.File 类的主要作用就是操作和获取这些元数据（例如 file.getName(), file.length()），而不是文件数据本身。

什么是文件

- 文件是操作系统封装底层复杂硬件操作（如磁盘块、扇区、磁道）后，提供给应用程序的一个**统一、简单的接口**。

- 程序员无需关心数据具体存储在硬盘的哪个磁道上，只需要通过“文件”这个抽象概念进行“读”和“写”操作。操作系统会负责完成所有底层的转换工作。这使得开发变得简单且通用。

什么是文件

- 根据文件中数据的组织格式，我们可以将文件分为两大类，这也直接决定了我们在Java中应该使用哪种流来处理它：
- 文本文件（Text File）

定义：文件中的字节内容遵循某种**字符编码规范**（如UTF-8, GBK, ASCII），可以被解析成人类可读的字符。

特点：

用文本编辑器（如记事本、VS Code）打开可以正常看到文字内容。

内容是由**行**组成的，每行通常以换行符（\n）或回车换行符（\r\n）结束。

常见扩展名：.txt, .java, .html, .xml, .csv, .json（注意：.csv和.json本质也是纯文本）。

在Java中如何处理：使用**字符流**（Reader, Writer及其子类，如 FileReader, BufferedWriter）。字符流会帮我们自动处理编码解码，让我们能以“字符”和“字符串”这个层面来操作文件，非常方便。

什么是文件

- 二进制文件（Binary File）

定义：文件中的字节内容**不代表字符**，而是代表其他类型的程序数据。这些字节的排列顺序有特殊的、非字符的含义。

特点：

用文本编辑器打开会看到一堆乱码。

必须由特定的软件按照特定的格式才能正确解读。

常见扩展名：

图片：.jpg, .png, .gif

音频：.mp3, .wav

视频：.mp4, .avi

可执行程序：.exe, .class

压缩包：.zip, .rar

文档：.docx, .xlsx（这些文件虽然存储了文本，但内部是压缩的、带有格式的复杂二进制结构，所以也属于二进制文件）

在Java中如何处理：使用**字节流**（InputStream, OutputStream及其子类，如 FileInputStream, FileOutputStream）。字节流直接操作原始的字节，不做任何转换，保证了二进制数据的原样读入和输出。

什么是文件

- 简单比喻

处理文本文件就像读一本小说。我们关心的是文字、句子和段落（字符和字符串）。

处理二进制文件就像分析一份DNA样本。我们关心的是里面各种化学物质的精确排列顺序（字节序列），而不是把它当成文字来读。

什么是文件

- 流的本质：

  它是一种抽象，屏蔽了底层硬件（磁盘、网卡等）的复杂性。程序员无需关心数据是如何在磁盘上磁道扇区中存储的，只需要从一个流中读取数据，或者向一个流中写入数据。

  流的操作是顺序的。通常只能顺序地从流中读取或写入数据，不能像数组一样随机访问（有特殊的RandomAccessFile类支持随机访问）。

什么是文件

- 流的分类：

按流向：数据从文件到程序是输入，从程序到文件是输出。判断的基准永远是程序（内存）。

InputStream / Reader -> 输入 -> 程序

OutputStream / Writer -> 程序 -> 输出

按数据类型：

字节流 (Byte Streams)：InputStream 和 OutputStream 是所有字节流的抽象基类。

操作单位：字节（byte，8位）。这意味着它们可以处理任何类型的数据，因为所有文件在底层都是字节的序列。

典型用途：处理图片（jpg, png）、视频（mp4）、音频（mp3）、压缩包（zip）、可执行文件（exe）等二进制文件。当然，它们也可以处理文本文件，但可能不够方便。

字符流 (Character Streams)：Reader 和 Writer 是所有字符流的抽象基类。

操作单位：字符（char，16位Unicode码元）。它们专门为处理文本数据而设计。

关键优势：它们内部包含了编码解码（encode/decode）的过程。当我们用Reader读取一个字节流时，它会根据指定的字符编码（如UTF-8, GBK）将字节转换成字符；当我们用Writer写入时，它会将字符转换回字节。这个过程对我们来说是透明的，极大简化了文本处理。

典型用途：处理 .txt, .csv, .log, .json等文本文件。

简单总结：用字节流处理一切，用字符流只处理文本。

什么是文件

特性	文本文件	二进制文件
本质	遵循字符编码的字节序列	具有特定格式的原始字节序列
可读性	用文本编辑器可读	用文本编辑器打开为乱码
组成单位	字符、行	字节、特定格式的数据块
Java处理流	字符流（Reader, Writer）	字节流（InputStream, OutputStream）
例子	.txt, .java, .html	.jpg, .mp3, .class, .docx

为什么需要读写文件

- 内存（RAM）的易失性：程序在内存中运行，其中创建的对象和数据在程序退出或计算机关闭后就会丢失。它们不是永久性的。

- 持久化存储（Persistence）的需求：我们需要将数据保存到非易失的存储介质（如硬盘、SSD）上，以便后续再次使用。这就是“持久化”。文件是操作系统提供的最基本的持久化存储单元。

- 实际应用场景：

  配置文件：服务器的连接信息、基础数据的配置。

  数据存储：HIS系统中的患者信息、LIS系统的检查检验信息。

  日志记录：记录程序运行的状态、错误信息，用于排查问题。

  数据交换：从CSV或Excel文件中导入/导出数据。

怎样读写文件

java.io.File

- **核心思想**：File对象代表的**不是一个文件本身，而是一个路径名的抽象表示**。我们可以把它理解为一个“文件指针”或“路径标签”。创建一个File对象并不会在磁盘上实际创建文件。
- **路径**：

  绝对路径：从盘符或根目录开始的完整路径。D:\test\file.txt (Windows) 或 /test/file.txt (Linux/Mac)。优点是**精确**，缺点是**移植性差**。

  相对路径：相对于当前工作目录（通常是项目根目录）的路径。"file.txt", "data/config.ini", "../parent.txt"。优点是**移植性好**，缺点是**需要明确当前工作目录**。
- **常用方法**：

  boolean exists(): 判断文件或目录是否存在。进行文件操作前**先判断是否存在**是个好习惯。

  boolean isFile() / boolean isDirectory(): 判断是文件还是目录。

  String getName(): 获取文件名。

  long length(): 获取文件长度（字节数）。

  boolean createNewFile(): **当且仅当该文件不存在时**，创建一个新的空文件。

  boolean mkdir(): 创建单级目录。boolean mkdirs(): 创建多级目录。

  File[] listFiles(): 列出目录下的所有文件和子目录。

怎样读写文件

- FileReader / FileWriter:

  角色：它们是基础字符流，充当了数据源（文件）和程序之间的“连接管道”。

  缺点：每次read()只读一个字符（每次write()只写一个字符），这意味着需要进行大量的底层系统调用，性能非常低。因此，在实际开发中，我们一般不会直接单独使用它们。

- BufferedReader / BufferedWriter:

  角色：它们是处理流/包装流。它们自身不直接连接到数据源，而是包裹在另一个流（如FileReader）之上。

  工作原理：它们内部有一个缓冲区（一个字符数组）。

  BufferedReader：会一次性从底层FileReader中读取一大块数据（比如8192个字符）到缓冲区。然后程序的read()或readLine()操作都是从这块缓冲区中获取数据，直到缓冲区空了才再去底层读取。这极大地减少了系统调用次数。

  BufferedWriter：程序的write()操作是先写入其内部的缓冲区，等缓冲区满了，再一次性写入底层FileWriter。

- 核心API:

  BufferedReader.readLine(): 这是读取文本文件最常用的方法。它读取一行文本（以换行符\n、\r或\r\n为标志），并返回一个String（不包含换行符）。如果读到文件末尾，返回null。

  BufferedWriter.newLine(): 写入一个跨平台的换行符。比直接写"\n"更可取。

怎样读写文件

- 标准用法：组合使用

```
// 标准的、高效的文本文件读取
try (BufferedReader br = new BufferedReader(new FileReader("input.txt"))) {
    String line;
    while ((line = br.readLine()) != null) {
        // 在这里处理每一行 line
    }
}
```

怎样读写文件

- FileInputStream / FileOutputStream:

角色：基础字节流，用于建立与二进制文件的“管道”。

核心方法：

int read(): 读取一个字节，返回0到255之间的int值。返回-1表示文件结束。

int read(byte[] b): 这是更高效、更常用的方法。尝试读取最多b.length个字节到字节数组b中，返回实际读取的字节数，返回-1表示文件结束。

void write(byte[] b, int off, int len): 将字节数组b中从off索引开始的len个字节写入流。

- 使用缓冲区的重要性:

和字符流一样，单字节读写效率极低。

标准做法：创建一个固定大小的字节数组（如1024或8192字节）作为缓冲区。循环读取，直到文件末尾。

怎样读写文件

- 示例代码

```
...
byte[] buffer = new byte[1024]; // 1KB的缓冲区
int length; // 每次实际读取的字节数
while ((length = fis.read(buffer)) != -1) { // 1. 尝试读取1KB数据到buffer
   fos.write(buffer, 0, length);              // 2. 将buffer中从0开始的前length个字节写出
}
```

为什么是write(buffer, 0, length)，而不是write(buffer)?

因为最后一次读取，很可能无法填满整个buffer（比如文件只剩200字节，但buffer大小是1024）。如果直接写入整个buffer，就会多写出824字节的垃圾数据。length记录了这次循环实际读到了多少有效数据，只写出这部分。

BufferedInputStream / BufferedOutputStream:

它们为字节流提供了内置的缓冲区功能，用法和BufferedReader/BufferedWriter类似，可以进一步提升性能。在很多情况下，使用它们比手动管理字节数组更方便，且效率相当。

怎样读写文件

- try-with-resources (TWR):

  语法：在try关键字后的括号中声明和初始化资源（所有实现了AutoCloseable接口的对象，如各种流）。

  优势：无论try块是正常结束还是因为异常中断，JVM都会自动调用这些资源的close()方法。这避免了繁琐的finally块和潜在的资源泄漏问题。

  这是现代Java代码的绝对标准，必须掌握。

- 传统的资源关闭方式：

  在finally块中手动关闭流。

  陷阱：关闭流本身也可能抛出异常，需要在finally内部再套一层try-catch。代码非常臃肿且容易出错。

- 字符编码问题：

  根源：FileReader和FileWriter的一个缺陷是：它们使用系统默认的字符编码。这意味着我们在中文Windows（编码GBK）上写的文件，放到Linux（编码通常是UTF-8）上用同样的程序读取，就会显示乱码。

  解决方案：放弃直接使用FileReader/FileWriter来指定编码。

  使用InputStreamReader和OutputStreamWriter这两个“转换流”。

  它们分别是Reader和Writer的子类，构造时需要传入一个底层的字节流和一个明确的字符编码。

怎样读写文件

- 推荐写法

```
// 明确指定UTF-8编码读取文本文件
try (BufferedReader br = new BufferedReader(
        new InputStreamReader(
                new FileInputStream("file.txt"), StandardCharsets.UTF_8))) {
    // ... 操作br
}

// 明确指定UTF-8编码写入文本文件
try (BufferedWriter bw = new BufferedWriter(
        new OutputStreamWriter(
                new FileOutputStream("file.txt"), StandardCharsets.UTF_8))) {
    // ... 操作bw
}
```

本章总结

- 一、 核心思想：流（Stream）

  是什么流：数据从一个地方（如文件）到另一个地方（如程序）的传输通道，抽象为“流”。数据像水流一样顺序流动。

  两大分类：

  按流向：输入流（Input）（数据源 -> 程序）、输出流（Output）（程序 -> 目的地）。

  按数据类型：

  字节流 (InputStream/OutputStream)：处理一切文件（图片、视频、文本等），操作单位是字节(byte)。

  字符流 (Reader/Writer)：专为文本文件设计，操作单位是字符(char)，内部自动处理编码转换。

- 二、 两大文件类型与对应的流选择

文件类型	特点	常用类	核心方法
文本文件 (.txt, .java, .html等)	内容是人类可读的字符	FileReader / FileWriter BufferedReader / BufferedWriter（推荐）	readLine() write(String s) newLine()
二进制文件 (.jpg, .mp3, .class等)	内容是字节，文本编辑器打开乱码	FileInputStream / FileOutputStream	read(byte[] b) write(byte[] b, int off, int len)

  黄金法则

  看到文本文件 -> 首选 BufferedReader/BufferedWriter。

  看到二进制文件 -> 直接用 FileInputStream/FileOutputStream 搭配字节数组缓冲区。

本章作业

- 综合运用字节流、字符流、异常处理等知识，完成一个具有基本文件操作功能的命令行程序。

作业要求:

1. 程序启动后，应显示一个菜单，供用户选择操作。
2. 所有流的操作必须使用 try-with-resources 语句确保正确关闭。
3. 处理文本文件时，必须使用缓冲流（BufferedReader/BufferedWriter）以提高效率。
4. 处理二进制文件时，必须使用字节数组缓冲区。
5. 必须进行必要的异常捕获和处理，给出用户友好的错误提示（如“文件未找到”），而不是让程序崩溃。

信创智能医疗系统研发课程体系

河南中医药大学信息技术学院（智能医疗行业学院）

河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所