别再用后缀判断文件类型了，来认识一下魔数头

引言

> 最近公司整改，招了个安全测试，安全测试的同事搞了个危险的文件，悄咪咪加了个.png的后缀，就丢到我们的生产服务器上面去了，这么勇，你敢信?
>
>
> 不管你信不信，事实他就是发生了，就问你怕不怕。
>
>
> 好了，这里也就暴露了一个很大的问题，我们的开发同事判断文件类型，是使用文件后缀来判断的，所以被直接跳过。咱来看看更加科学的识别方式。

一、认识魔数

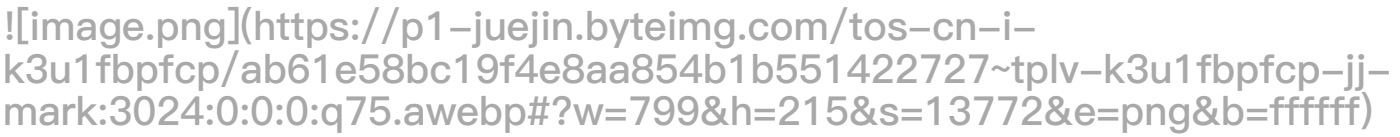
魔数：也被称为签名或文件签名，是一种用于识别文件类型和格式的短序列字节。它们通常位于文件的开头，并被设计为易于识别，以便软件可以快速确定文件是否是它所支持的格式。

魔数是一种简单的识别机制，它由一系列字节组成，这些字节在特定的文件格式中是唯一的。当一个程序打开一个文件时，它会检查文件的开始处是否包含这些特定的字节。如果找到，程序就认为文件是该格式的，并按照相应的规则进行解析和处理。

二、文件类型检测的原理

> 文件类型检测通常基于文件的“魔数”（magic number），也称为签名或文件签名。魔数是文件开头的字节序列，用于标识文件格式。以下是文件类型检测的原理和步骤：

2.1 文件头的读取方法

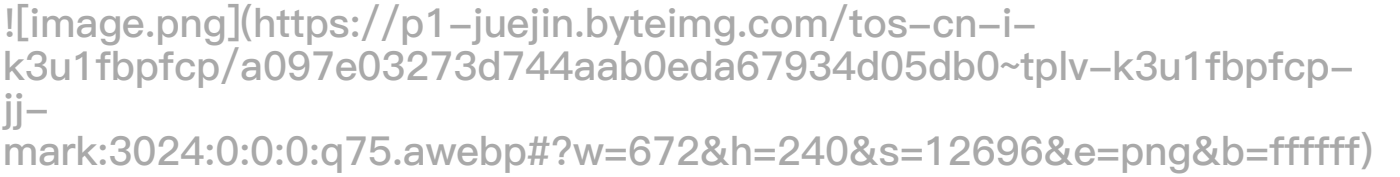


打开文件：首先，需要以二进制模式打开文件，以便能够读取文件的原始字节。

读取字节：接着，读取文件开头的一定数量的字节（通常是前几个字节）。

关闭文件：读取完成后，关闭文件以释放资源。

2.2 如何通过文件头识别文件类型



比较魔数：将读取的字节与已知的文件类型魔数进行比较。

匹配类型：如果字节序列与某个文件类型的魔数匹配，则可以确定文件类型。

处理异常：如果字节序列与任何已知魔数都不匹配，可能需要进一步的分析或返回未知文件类型。

三、Java实现文件类型检测

当需要通过文件头（魔数头）判断文件类型时，可以按照以下文字描述的流程进行实现：

```
```

```
graph LR
F[打开文件] --> B[读取文件头]
B --> C[判断文件类型]
C --> D[比较文件头]
D --> E[输出文件类型]

style B fill:#FFC0CB,stroke:#FFC0CB,stroke-width:2px
style C fill:#FFA07A,stroke:#FFA07A,stroke-width:2px
style D fill:#FFFFE0,stroke:#FFFFE0,stroke-width:2px
style E fill:#98FB98,stroke:#98FB98,stroke-width:2px
style F fill:#B2FFFF,stroke:#B2FFFF,stroke-width:2px
```

1. **打开文件**：使用文件输入流（FileInputStream）打开待判断类型的文件。
2. **读取文件头**：从文件中读取一定长度的字节数据作为文件头。通常，文件头的长度为固定的几个字节，一般是 2-8 个字节。
3. **判断文件类型**：根据不同文件类型的魔数头进行判断。魔数头是文件中特定位置的字节序列，用于标识文件类型。每种文件类型都有不同的魔数头。
4. **比较文件头**：将读取到的文件头与已知文件类型的魔数头进行比较。如果匹配成功，则确定文件类型。
5. **输出文件类型**：根据匹配的文件类型，输出相应的文件类型描述信息。

3.1 具体实现方法

3.1.1 定义枚举类

```
/**
 * 文件类型魔数枚举
 * 使用场景：用于判断文件类型
 * 使用方法：FileUtils.isFileType(new FileInputStream(file), FileTypeEnum.XLSX)
 *
 * @author bamboo panda
 * @version 1.0
 * @date 2024/5/23 19:37
 */
public enum FileTypeEnum {
    /**
     * JPEG
     */
    JPEG("JPEG", "FFD8FF"),
```

```
/**
 * PNG
 */
PNG(”PNG”, ”89504E47”),

/**
 * GIF
 */
GIF(”GIF”, ”47494638”),

/**
 * TIFF
 */
TIFF(”TIFF”, ”49492A00”),

/**
 * Windows bitmap
 */
BMP(”BMP”, ”424D”),

/**
 * CAD
 */
DWG(”DWG”, ”41433130”),

/**
 * Adobe photoshop
 */
PSD(”PSD”, ”38425053”),

/**
 * Rich Text Format
 */
RTF(”RTF”, ”7B5C727466”),

/**
 * XML
 */
XML(”XML”, ”3C3F786D6C”),

/**
 * HTML
 */
HTML(”HTML”, ”68746D6C3E”),

/**
 * Outlook Express
```

```
 */
DBX(”DBX”, ”CFAD12FEC5FD746F ”),

/**
 * Outlook
 */
PST(”PST”, ”2142444E”),

/**
 * doc;xls;dot;ppt;xla;ppa;pps;pot;msi;sdw;db
 */
OLE2(”OLE2”, ”0xD0CF11E0A1B11AE1”),

/**
 * Microsoft Word/Excel
 */
XLS_DOC(”XLS_DOC”, ”D0CF11E0”),

/**
 * Microsoft Access
 */
MDB(”MDB”, ”5374616E64617264204A”),

/**
 * Word Perfect
 */
WPB(”WPB”, ”FF575043”),

/**
 * Postscript
 */
EPS_PS(”EPS_PS”, ”252150532D41646F6265”),

/**
 * Adobe Acrobat
 */
PDF(”PDF”, ”255044462D312E”),

/**
 * Windows Password
 */
PWL(”PWL”, ”E3828596”),

/**
 * ZIP Archive
 */
ZIP(”ZIP”, ”504B0304”),
```

```
/**
 * ARAR Archive
 */
RAR("RAR", "52617221"),

/**
 * WAVE
 */
WAV("WAV", "57415645"),

/**
 * AVI
 */
AVI("AVI", "41564920"),

/**
 * Real Audio
 */
RAM("RAM", "2E7261FD"),

/**
 * Real Media
 */
RM("RM", "2E524D46"),

/**
 * Quicktime
 */
MOV("MOV", "6D6F6F76"),

/**
 * Windows Media
 */
ASF("ASF", "3026B2758E66CF11"),

/**
 * MIDI
 */
MID("MID", "4D546864"),
/**
 * xlsx
 */
XLSX("XLSX", "504B0304"),
/**
 * xls
 */
XLS("XLS", "D0CF11E0A1B11AE1");
```

```
    private String key;
    private String value;

    FileTypeEnum(String key, String value) {
        this.key = key;
        this.value = value;
    }

    public String getValue() {
        return value;
    }

    public String getKey() {
        return key;
    }
}
```

3.1.2 文件类型判断工具类

```
import java.io.IOException;
import java.io.InputStream;

/**
 * 文件类型判断工具类
 *
 * @author bamboo panda
 * @version 1.0
 * @date 2024/5/23 19:38
 */
public class FileTypeUtils {

    /**
     * 获取文件头
     *
     * @param inputStream 输入流
     * @return 16 进制的文件投信息
     * @throws IOException io异常
     */
    private static String getFileHeader(InputStream inputStream) throws
IOException {
        byte[] b = new byte[28];
        inputStream.read(b, 0, 28);
        inputStream.close();
```

```
        return bytes2hex(b);
    }

    /**
     * 将字节数组转换成16进制字符串
     *
     * @param src 文件字节数组
     * @return 16进制字符串
     */
    private static String bytes2hex(byte[] src) {
        StringBuilder stringBuilder = new StringBuilder("");
        if (src == null || src.length <= 0) {
            return null;
        }
        for (byte b : src) {
            int v = b & 0xFF;
            String hv = Integer.toHexString(v);
            if (hv.length() < 2) {
                stringBuilder.append(0);
            }
            stringBuilder.append(hv);
        }
        return stringBuilder.toString();
    }

    /**
     * 判断指定输入流是否是指定文件格式
     *
     * @param inputStream  输入流
     * @param fileTypeEnum 文件格式枚举
     * @return true 是； false 否
     * @throws IOException io异常
     */
    public static boolean isFileType(InputStream inputStream,
FileTypeEnum fileTypeEnum) throws IOException {
        if (null == inputStream) {
            return false;
        }
        String fileHeader = getFileHeader(inputStream);
        return
fileHeader.toUpperCase().startsWith(fileTypeEnum.getValue());
    }

}

```

3.1.3 测试方法

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

/**
 * 测试：判断文件是否是excel
 *
 * @author bamboo panda
 * @version 1.0
 * @date 2024/5/23 19:33
 */
public class Test {

    public static void main(String[] args) {
        File file = new File("C:\Users\Admin\Desktop\temp\Import
file.xlsx");
        try (FileInputStream fileInputStream = new FileInputStream(file)) {
            if (!FileTypeUtils.isFileType(fileInputStream, FileTypeEnum.XLSX)
|| !FileTypeUtils.isFileType(fileInputStream, FileTypeEnum.XLS)) {
                System.out.println(true);
            } else {
                System.out.println(false);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

四、其它注意事项

在处理文件类型检测和数据保护时，安全性和隐私是两个非常重要的考虑因素。以下是一些相关的安全性问题和最佳实践：

4.1 魔数检测的安全性问题

```
graph LR
F(文件类型判断)
B(魔数检测的安全性问题)
C(误报和漏报)
D(恶意文件伪装)
E(更新和维护)

F ---> B
B ---> C
B ---> D
B ---> E

style B fill:#FFC0CB,stroke:#FFC0CB,stroke-width:2px
style C fill:#FFA07A,stroke:#FFA07A,stroke-width:2px
style D fill:#FFFFE0,stroke:#FFFFE0,stroke-width:2px
style E fill:#98FB98,stroke:#98FB98,stroke-width:2px
style F fill:#B2FFFF,stroke:#B2FFFF,stroke-width:2px
```

1. **误报和漏报**：魔数检测可能因为文件损坏或不完整而产生误报或漏报。一些恶意软件可能会模仿合法文件的魔数来逃避检测。
2. **恶意文件伪装**：攻击者可能故意在文件中嵌入合法的魔数，使得恶意文件看起来像是合法的文件类型。
3. **更新和维护**：随着新文件类型的出现和旧文件类型的淘汰，魔数列表需要定期更新，否则检测系统可能会变得不准确或过时。

4.2 数据保护和隐私的最佳实践

```
graph LR
I(文件类型判断)

A(数据保护和隐私的最佳实践)

G(最小权限原则)
B(数据加密)
C(安全的数据传输)
D(访问控制)
E(定期审计和测试)
F(数据最小化)
```

I ---> A
A ---> B
A ---> C
A ---> D
A ---> E
A ---> F
A ---> G

style B fill:#FFC0CB,stroke:#FFC0CB,stroke-width:2px
style C fill:#FFA07A,stroke:#FFA07A,stroke-width:2px
style D fill:#FFFFE0,stroke:#FFFFE0,stroke-width:2px
style E fill:#98FB98,stroke:#98FB98,stroke-width:2px
style F fill:#B2FFFF,stroke:#B2FFFF,stroke-width:2px
style G fill:#ADD8E6,stroke:#ADD8E6,stroke-width:2px
style A fill:#E6E6FA,stroke:#E6E6FA,stroke-width:2px
style I fill:#EEDD82,stroke:#EEDD82,stroke-width:2px

```

1. **最小权限原则**：确保应用程序只请求执行其功能所必需的权限，不要求额外的权限。
2. **数据加密**：对敏感数据进行加密，无论是在传输中还是存储时，都应使用强加密标准。
3. **安全的数据传输**：使用安全的协议（如HTTPS）来保护数据在网络中的传输。
4. **访问控制**：实施严格的访问控制措施，确保只有授权用户才能访问敏感数据。
5. **定期审计和测试**：定期进行安全审计和渗透测试，以发现和修复潜在的安全漏洞。
6. **数据最小化**：只收集完成服务所必需的最少数据量，避免收集不必要的个人信息。

### 4.3 魔数的局限性和风险

魔数判断文件类型是一种常用的方法，但也存在一些局限性和风险，包括以下几点：

```
graph LR
I(文件类型判断)

A(魔数的局限性和风险)

B(可伪造性)
C(文件类型扩展性)
D(文件损坏或篡改)
E(多重文件类型)
F(文件类型模糊性)

I ---> A
A ---> B
A ---> C
A ---> D
A ---> E
A ---> F

style B fill:#FFC0CB,stroke:#FFC0CB,stroke-width:2px
style C fill:#FFA07A,stroke:#FFA07A,stroke-width:2px
style D fill:#FFFFE0,stroke:#FFFFE0,stroke-width:2px
style E fill:#98FB98,stroke:#98FB98,stroke-width:2px
style F fill:#B2FFFF,stroke:#B2FFFF,stroke-width:2px
style A fill:#E6E6FA,stroke:#E6E6FA,stroke-width:2px
style I fill:#EEDD82,stroke:#EEDD82,stroke-width:2px

```

1. **可伪造性**：魔数头是文件中的特定字节序列，攻击者可以通过修改文件的魔数头来伪装文件类型。这可能导致误判文件类型或绕过文件类型检测。
2. **文件类型扩展性**：随着新的文件类型的出现，魔数头的定义可能需要不断更新，以适应新的文件类型。如果应用程序不及时更新对新文件类型的判断逻辑，可能无法正确识别新的文件类型。
3. **文件损坏或篡改**：如果文件的魔数头部分被损坏或篡改，可能导致无法正确判断文件类型，或者将文件错误地归类为不正确的类型。
4. **多重文件类型**：某些文件可能具有多重文件类型，即使使用魔数头判断了其中一种类型，也可能存在其他类型。这可能导致文件类型的混淆和判断的不准确性。
5. **文件类型模糊性**：某些文件类型可能具有相似或相同的魔数头，这可能导致在这些类型之间进行区分时出现困难。这可能增加了误判文件类型的风险。

## 五、总结

好了，到这里魔数怎么用的就说明白了。

魔数的广泛的应用在在文件类型检测中。魔数是文件开头的特定字节序列，帮助软件快速识别文件格式。
```

然而，魔数检测存在安全性问题，如误报、恶意伪装等，需定期更新魔数库。此外，应用魔数检测时要考虑文件损坏、多重类型等局限性，结合实际情况采取综合措施，如数据加密、访问控制等，确保安全性和准确性。

> 希望本文对您有所帮助。如果有任何错误或建议，请随时指正和提出。
>
>
> 同时，如果您觉得这篇文章有价值，请考虑点赞和收藏。这将激励我进一步改进和创作更多有用的内容。
>
>
> 感谢您的支持和理解！

原文链接: https://juejin.cn/post/7372100124636381194