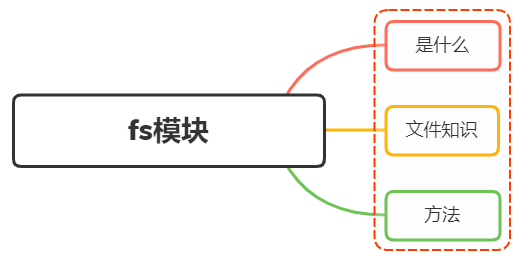
# 面试官：说说对 Node 中的 fs模块的理解? 有哪些常用方法



## 一、是什么

fs（filesystem），该模块提供本地文件的读写能力，基本上是POSIX文件操作命令的简单包装

可以说，所有与文件的操作都是通过fs核心模块实现

导入模块如下：

const fs = require('fs');

这个模块对所有文件系统操作提供异步（不具有sync 后缀）和同步（具有 sync 后缀）两种操作方式，而供开发者选择

### 二、文件知识

在计算机中有关于文件的知识：

* 权限位 mode
* 标识位 flag
* 文件描述为 fd

### 权限位 mode



针对文件所有者、文件所属组、其他用户进行权限分配，其中类型又分成读、写和执行，具备权限位4、2、1，不具备权限为0

如在linux查看文件权限位：

drwxr-xr-x 1 PandaShen 197121 0 Jun 28 14:41 core  
-rw-r--r-- 1 PandaShen 197121 293 Jun 23 17:44 index.md

在开头前十位中，d为文件夹，-为文件，后九位就代表当前用户、用户所属组和其他用户的权限位，按每三位划分，分别代表读（r）、写（w）和执行（x），- 代表没有当前位对应的权限

### 标识位

标识位代表着对文件的操作方式，如可读、可写、即可读又可写等等，如下表所示：

|  |  |
| --- | --- |
| 符号 | 含义 |
| r | 读取文件，如果文件不存在则抛出异常。 |
| r+ | 读取并写入文件，如果文件不存在则抛出异常。 |
| rs | 读取并写入文件，指示操作系统绕开本地文件系统缓存。 |
| w | 写入文件，文件不存在会被创建，存在则清空后写入。 |
| wx | 写入文件，排它方式打开。 |
| w+ | 读取并写入文件，文件不存在则创建文件，存在则清空后写入。 |
| wx+ | 和 w+ 类似，排他方式打开。 |
| a | 追加写入，文件不存在则创建文件。 |
| ax | 与 a 类似，排他方式打开。 |
| a+ | 读取并追加写入，不存在则创建。 |
| ax+ | 与 a+ 类似，排他方式打开。 |

### 文件描述为 fd

操作系统会为每个打开的文件分配一个名为文件描述符的数值标识，文件操作使用这些文件描述符来识别与追踪每个特定的文件

Window系统使用了一个不同但概念类似的机制来追踪资源，为方便用户，NodeJS抽象了不同操作系统间的差异，为所有打开的文件分配了数值的文件描述符

在 NodeJS中，每操作一个文件，文件描述符是递增的，文件描述符一般从 3 开始，因为前面有 0、1、2三个比较特殊的描述符，分别代表 process.stdin（标准输入）、process.stdout（标准输出）和 process.stderr（错误输出）

## 三、方法

下面针对fs模块常用的方法进行展开：

* 文件读取
* 文件写入
* 文件追加写入
* 文件拷贝
* 创建目录

### 文件读取

#### fs.readFileSync

同步读取，参数如下：

* 第一个参数为读取文件的路径或文件描述符
* 第二个参数为 options，默认值为 null，其中有 encoding（编码，默认为 null）和 flag（标识位，默认为 r），也可直接传入 encoding

结果为返回文件的内容

const fs = require("fs");  
  
let buf = fs.readFileSync("1.txt");  
let data = fs.readFileSync("1.txt", "utf8");  
  
console.log(buf); // <Buffer 48 65 6c 6c 6f>  
console.log(data); // Hello

#### fs.readFile

异步读取方法 readFile 与 readFileSync 的前两个参数相同，最后一个参数为回调函数，函数内有两个参数 err（错误）和 data（数据），该方法没有返回值，回调函数在读取文件成功后执行

const fs = require("fs");  
  
fs.readFile("1.txt", "utf8", (err, data) => {  
 if(!err){  
 console.log(data); // Hello  
 }  
});

### 文件写入

#### writeFileSync

同步写入，有三个参数：

* 第一个参数为写入文件的路径或文件描述符
* 第二个参数为写入的数据，类型为 String 或 Buffer
* 第三个参数为 options，默认值为 null，其中有 encoding（编码，默认为 utf8）、 flag（标识位，默认为 w）和 mode（权限位，默认为 0o666），也可直接传入 encoding

const fs = require("fs");  
  
fs.writeFileSync("2.txt", "Hello world");  
let data = fs.readFileSync("2.txt", "utf8");  
  
console.log(data); // Hello world

#### writeFile

异步写入，writeFile 与 writeFileSync 的前三个参数相同，最后一个参数为回调函数，函数内有一个参数 err（错误），回调函数在文件写入数据成功后执行

const fs = require("fs");  
  
fs.writeFile("2.txt", "Hello world", err => {  
 if (!err) {  
 fs.readFile("2.txt", "utf8", (err, data) => {  
 console.log(data); // Hello world  
 });  
 }  
});

### 文件追加写入

#### appendFileSync

参数如下：

* 第一个参数为写入文件的路径或文件描述符
* 第二个参数为写入的数据，类型为 String 或 Buffer
* 第三个参数为 options，默认值为 null，其中有 encoding（编码，默认为 utf8）、 flag（标识位，默认为 a）和 mode（权限位，默认为 0o666），也可直接传入 encoding

const fs = require("fs");  
  
fs.appendFileSync("3.txt", " world");  
let data = fs.readFileSync("3.txt", "utf8");

#### appendFile

异步追加写入方法 appendFile 与 appendFileSync 的前三个参数相同，最后一个参数为回调函数，函数内有一个参数 err（错误），回调函数在文件追加写入数据成功后执行

const fs = require("fs");  
  
fs.appendFile("3.txt", " world", err => {  
 if (!err) {  
 fs.readFile("3.txt", "utf8", (err, data) => {  
 console.log(data); // Hello world  
 });  
 }  
});

### 文件拷贝

#### copyFileSync

同步拷贝

const fs = require("fs");  
  
fs.copyFileSync("3.txt", "4.txt");  
let data = fs.readFileSync("4.txt", "utf8");  
  
console.log(data); // Hello world

#### copyFile

异步拷贝

const fs = require("fs");  
  
fs.copyFile("3.txt", "4.txt", () => {  
 fs.readFile("4.txt", "utf8", (err, data) => {  
 console.log(data); // Hello world  
 });  
});

### 创建目录

#### mkdirSync

同步创建，参数为一个目录的路径，没有返回值，在创建目录的过程中，必须保证传入的路径前面的文件目录都存在，否则会抛出异常

// 假设已经有了 a 文件夹和 a 下的 b 文件夹  
fs.mkdirSync("a/b/c")

#### mkdir

异步创建，第二个参数为回调函数

fs.mkdir("a/b/c", err => {  
 if (!err) console.log("创建成功");  
});

## 参考文献

* http://nodejs.cn/api/fs.html
* https://segmentfault.com/a/1190000019913303