【node】同步读取readFileSync和异步读取readFile的区别案例

先结论：异步读取文件的时候由操作系统在后台进行读取，不会阻碍下面的代码执行。同步读取的时候会阻碍下面的代码执行。

再案例：

test.js

```
var fs = require('fs');                 //获取fs模块
console.log('start async read');         //异步读取开始
fs.readFile('test2.js',function(error,date){     //读取文件，回调函数第一个参数表示错误信息，第二个参数为读取的文本内容
    if(error){
        console.log(error);
    }else{
        console.log('end async read');     //异步读取结束
    }
});


console.log('start sync read'); //同步读取开始
var date = fs.readFileSync('log.txt','utf-8');
console.log('end sync read');     //同步读取结束
```

结果：

```
start async read
start sync read
end sync read
end async read

```

看出是按这个顺序执行的：异步读取开始---------->同步读取开始---------->同步读取结束---------->异步读取结束

异步读取没有结束，同步读取就完成了。

结论：异步读取没有阻塞下面代码的执行。

调换一下代码就更明显了

```
var fs = require('fs');                 //获取fs模块

console.log('start sync read'); //同步读取开始
var date = fs.readFileSync('log.txt','utf-8');
console.log('end sync read');     //同步读取结束


console.log('start async read');         //异步读取开始
fs.readFile('test2.js',function(error,date){     //读取文件，回调函数第一个参数表示错误信息，第二个参数为读取的文本内容
    if(error){
        console.log(error);
    }else{
        console.log('end async read');     //异步读取结束
    }
});
```

结果：

```
start sync read
end sync read
start async read
end async read

```

看出是按这个顺序执行的：同步读取开始---------->同步读取结束---------->异步读取开始---------->异步读取结束

结论：同步读取阻塞下面代码执行。