# 理解buffer（pdf155张）

6.1 buffer结构

6.1.1 buffer模块结构

Buffer是C++和js的组合，buffer占用的内存不是v8内存，属于栈外内存。

6.1.2 buffer对象

Buffer对象类似数组，它的元素为16进制的两位数，表示0-255内一个。其中中文站3个元素，字母和符号占1一个元素。可以直接对buffer的其中一位进行赋值，如果赋值小数则忽略小数部分，赋值小于0则加256直至满足0-255，如果赋值超过255则－256直到满足0-255.

6.1.3 buffer内存分配

Buffer的内存分配采用slab分配机制。简而言之slab就是一块申请好的固定大小的内存区域。内存有三种状态：

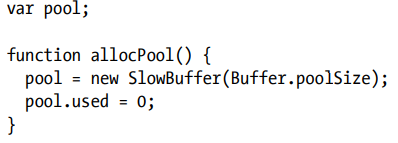
* Full：完成分配状态。
* Partial：部分分配状态。
* Empty：没有被分配状态。

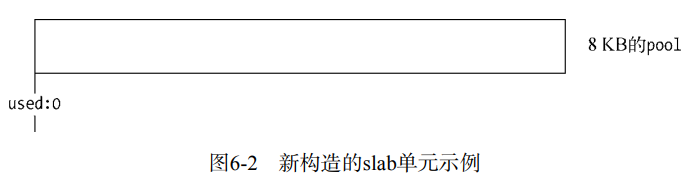
在node里以8kb来区分Buffer是大对象还是小对象。

Buffer.poolSize = 8 \* 1024;

分配小buffer对象

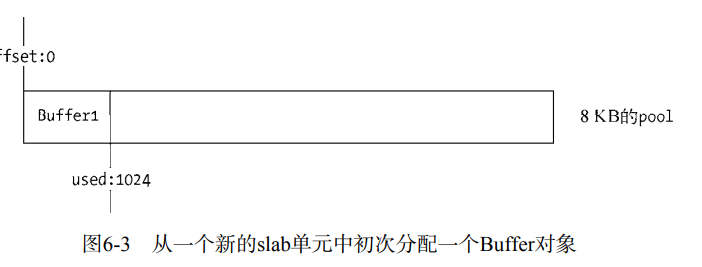
少于8kb的buffer都按照小buffer对象来分配。Node主要使用一个局部变量pool作为中间处理对象，所以slab对象都指向它。





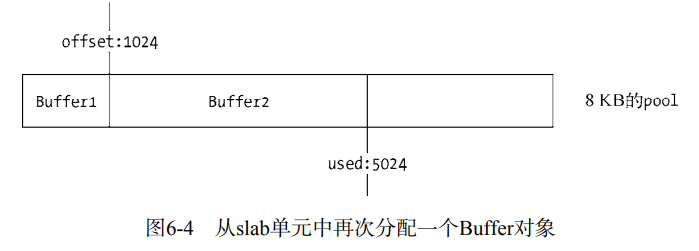
构建一个新的slab单元，此时slab单元处于empty状态。

构建小buffer对象时，new Buffer(1024); 去检测pool对象，如果pool没有被创建，则创建一个slab对象指向它，buffer对象的parent属性指向这个slab对象，offset属性记录从什么地方开始分配。



这时候slab状态为partial。

再次创建一个buffr对象时，入股剩余空间足够，则使用剩余空间，如果不够则创建一个新的slab。new Buffer(3000);



分配大buffer对象

如果超过8kb的buffer对象，直接分配一个SlowBuffer对象作为slab。



6.2.2 buffer转字符串

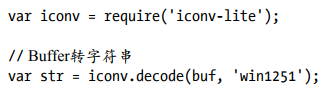
Buffer对象的toString方法可以将buffer转成字符串，其中开始和结束是buffer中的位置，即汉字占3个长度。



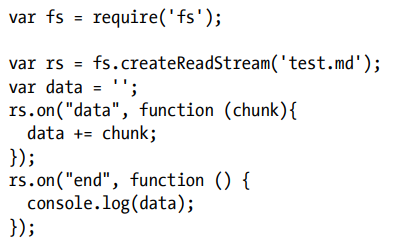
6.2.3 buffer不支持的编码类型

Buffer.isEncoding(encoding)可以用来判断buffer是否支持这类编码。中国常用的GBK，GB2312，BIG-5都不支持。

对于不支持的编码可以使用iconv-lite模块。

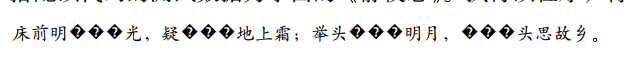
 

6.3 buffer的拼接

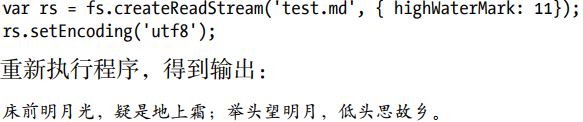


[Demo1](demo1.js)。其中chunk对象就是buffer对象，一但输入流中有宽字节编码则会看到乱码，那么问题多半是data += chunk;代码中包含了data = data.toString() + chunk.toString();可以设置文件可独流，每次读取buffer的长度。

var rs = fs.createReadStream('test.md', {highWaterMark: 11}); 得到以下结果，产生结果即中文站3个长度，11刚好不是3的倍数。



6.3.2 setEncoding()与string\_decoder()

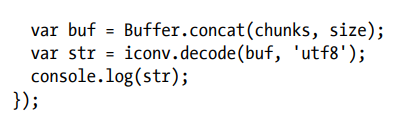
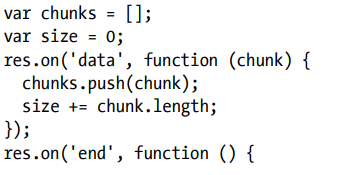


每次data的buffer长度不变，但是调用了setEncoding（）之后，可读流对象内部设置了一个decoder对象，每次用decoder对象解析buffer对象之后再传递给调用者。Decoder又是StringDecoder实例。

再设置uft-8编码之后，StringDecoder只处理前9个，剩下2个字节保存在StringDecoder内部，第二次会将这2个与其他11个字节组合再进行转码。

string\_decoder目前只能处理UTF-8、Base64和UCS-2/UTF-16LE三种编码。

6.3.3 正确拼接buffer方式



记录下每一次buffer的内容和总的长度，最后将多个小buffer转成大buffer。对于buffer不支持的编码可以使用iconv模块或iconv-lite模块。[Demo2](demo2.js)

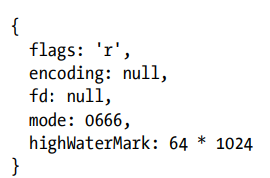
6.4 buffer与性能

Buffer广泛运用到文件IO和网络IO中。在应用中我们用的是字符串，一但需要进行网络传输，就要将字符串转换成buffer的二进制数据。提升buffer的性能可以很大得提高网络吞吐量。

使用buffer代替string来回复网络请求，能提高将近一倍的性能。

* 文件读取

fs.createReadStream()的工作方式是在内存中准备一个buffer，然后用fs.read()逐步将硬盘中的字节复制到buffer中，读完后通过Buffer对象的slice()方法取出部分数据作为一个小的buffer，经过data事件传给调用方。如果buffer内存不够则重新分配一个。fs.createReadStream(path, opts)第二个参数可以设置多个属性。





highWaterMark值越大读取速度越快。