用 string 存取二进制数据

STL 的 string 很强大，用起来也感觉很舒服，这段时间在代码中涉及到了用 string 存取二进制数据的问题，这里记录一下，以供以后参考。

首先提一下 STL 中 string 的参考资料：http://www.cplusplus.com/reference/string/string/ ，不懂的朋友可以看下。

在数据传输中，二进制数据的 buffer 一般用系统预设的大数组进行存储，而不是 STL 的 string 等，比如：

const int max_length = 1024 * 1024;

unsigned char data[max_length];

因为二进制数据中可能会包含 0x00（即：'\0'），刚好是字符串结束标志……

如果我们的代码是如下写的：

char data[max_length];

size_t length = sockClient.read_some(boost::asio::buffer(data), ec);

string strData(data);

我只能说，这个处理字符串应该没问题，如果是二进制的话，会被 string 的构造函数给截断一部分，导致 strData 和 data 的数据不一致。

其实一个简单的 demo 就可以说明问题，比如如下代码：

```
#include <string>
#include <iostream>
using namespace std;

int main()
{
	char data[] = {'A','b',0x00,'c','d'};
	string str1(data),str2(data,sizeof(data));
	cout<<str1<<endl;
	cout<<str1.size()<<endl;
	cout<<str2<<endl;
	cout<<str2.size()<<endl;
	return 0;
}
```

运行效果：

Ab

2

Abcd

5

从运行结果不难发现，采用 str2 的那种方式就可以保证 string 中的数据和原始 data 中的数据一致。这是因为采用不同的构造函数不同，导致结构完全不一样，这个可以从我前面给出的

网址中去查看具体的构造函数说明加以理解。这里我们回到前面的那个问题，如果想保存二进制的话，我们应该如下操作：

```
char data[max_length];
size_t length = sockClient.read_some(boost::asio::buffer(data), ec);
string strData(data,length);
```

如果要取出数据的话，也简单（这个还以 socket 数据收发为例）：

```
.....
// deal with strData
.....
boost::asio::write(sockClient, boost::asio::buffer(strData.c_str(),strData.length()));
```

这里的 strData.c_str()即为数据，strData.length()即为要发送的数据长度（当然也可以使用 strData.size()来操作）。

当然，我这里用 string 来存取二进制数据，也只是为了操作方便，感觉这个不是太好，应该会有很多朋友不提倡这种做法的，这里提供一个思路，大家觉得好就采用，觉得不好就一笑了之，呵呵……

好，就这些了，希望对你有帮助。