**多国语言处理说明**

1. **参考**

参考出处：[**http://www.cublog.cn/u3/115019/showart\_2247318.html**](http://www.cublog.cn/u3/115019/showart_2247318.html)

**多国语言和字符集关系：**

字符集是对语言文字用进行特定的编码方式产生的字符。常用的编码方式有Unicode（UCS-2）、GB2312、UTF8等编码方式等。

字符集与C++中多字符设置选项并无直接的关系。C++设置Unicode 说明会通过Unicode编码方式对文字进行解码。多字节亦然。

**注意问题：**

1. **编码选择**：多国语言环境的编程，以使用 UTF 编码为原则，减少字符集转换。
2. **string 并不包含编码信息，但是编码确定了 string 的二进制内容。**
3. **读写一致**：读入时使用的字符集要与写出时使用的一致。如果不需要改变字符串内容，仅仅是将字符串读入、再写出，建议不要调整任何字符集——即使程序使用的系统默认字符集 A 与文件的实际编码 B 不符合，写出的字符串依然会是正确的 B 编码。
4. **读入已知**：对于必须处理、解析或显示的字符串，从文件读入时必须知道它的编码，避免处理字符串的代码简单使用系统默认字符集；即便对于程序从系统中收集到的内存字符串，也应知道其符合的编码格式——一般为系统默认字符集。
5. **避免直接使用 Unicode**：这里是说将非 ASCII 编码的 16 进制或者 10 进制数值用 &# 与 ; 包含起来的使用方式，例如将中文“一”写成“&#4e00;”。这种方法的实质是 Unicode 编码直接写入文件。这不仅会降低代码的通用性、输出文件的可读性，处理起来也很困难。比如法文字符在其他字符集中是大于 80H 的单字节字符，程序同时要支持中文的时候，很有可能会将多字节的中文字符错误割裂。
6. **避免陷入直接的字符集编程**：国际化、本地化的工具已经比较成熟，非纯粹做编码转换的程序员没有必要自己去处理不同编码表的映射转换问题。
7. **Unicode/UTF8 并不能解决一切乱码问题**：Unicode 可以说是将世界语言统一起来的一套编码。但是这并不意味着在一个系统中可以正常显示的按照 UTF8 编码的文件，在另一个系统中也可以正常显示。例如，在中文的 UTF8 编码或者 Unicode 编码在没有东亚语言包支持的法文系统中，依然是不可识别的乱码——尽管 UTF8、Unicode 它们都支持。
8. **项目中采用的多国语言处理方式**
9. **凡是涉及到需要显示在客户端或者服务器端的文字一律用Unicode的编码方式。包括数据库中存储的文字。**
10. **凡是编码需要转换。先转换成Unicode编码，再转化成想要的编码。例如下图：**

