**长江大学**

**课程小论文**

**2017 学年 第二 学期**

**学 号：\_\_\_\_\_\_\_201503484\_\_\_\_\_\_\_\_\_**

**姓 名：\_\_\_\_\_\_\_\_ 马胜超\_\_\_\_\_\_\_\_\_\_**

**所 属：\_\_\_\_\_\_\_\_软工1501\_\_\_\_\_\_\_\_**

**评 语：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**xml小课题作业**

马胜超

**摘要：GB2312 BIG5 UTF8/16 UNICODE 几个编码集的来历、产生原因、解决的问题、字节数、几种编码集之间能否相互转换，能否兼容**

**关键词：GB2312 BIG5 UTF8/16 UNICODE**

**一、背景介绍**

随着计算机的发展，最开始的ASCII编码已经不能够满足各国的需求，人们需要对各国的大量字符进行编码。因而出现了针对简体字的字符集GB2311，针对繁体字的字符集Big5。然而多种编码方式的出现，也产生了很多的问题。比如两种不同的字符对应的二进制的编码是一样的，这样计算机无法识别。为了解决这种问题出现了Unicode字符集来统一规定，既要满足字符集之间的兼容。在Unicode字符集中，解决了第一层次字符与二进制数的对应问题，第二层次二进制数在电脑中的存储方式又有UTF-8编码和UTF-16编码两种不同。我们了解这几种编码方式，既可以学习别人的思考方式，编码特点，也为我们学习其他的计算机知识奠定更好的基础。

**二、GB2312**

**1.来历**

**GB 2312** （又称 **GB 2312–80** ）是中华人民共和国国家标准简体中文字符集，全称《**信息交换用汉字编码字符集·基本集**》，又称GB0，由中国国家标准总局发布，1981年5月1日实施。GB 2312编码通行于中国大陆；新加坡等地也采用此编码。中国大陆几乎所有的中文系统和国际化的软件都支持GB 2312。

GB 2312标准共收录6763个汉字，其中一级汉字3755个，二级汉字3008个；同时收录了包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母在内的682个字符。

**2.产生原因**

GB2312编码的产生是为了解决简体中文进行编码的问题，它收录了几乎所有的简体汉字以及一些非汉字符号。（GBK编码是对GB2312编码的扩展，收录中日韩文字以及Big5编码的繁体字。GB18030编码又是对GBK编码的扩充，在此基础上收录了多种我国少数民族的文字。

**3.解决问题**

GB 2312的出现，基本满足了汉字的计算机处理需要，它所收录的汉字已经覆盖中国大陆99.75%的使用频率。但对于人名、古汉语等方面出现的罕用字和繁体字，GB2312不能处理，因此后来GBK和GB18030汉字字符集相继出现以解决这些问题。

**4.字节数**

在使用GB 2312的程序通常采用[EUC](https://zh.wikipedia.org/wiki/EUC)储存方法，以便兼容于[ASCII](https://zh.wikipedia.org/wiki/ASCII)。这种格式称为EUC-CN。[浏览器](https://zh.wikipedia.org/wiki/%E7%BD%91%E9%A1%B5%E6%B5%8F%E8%A7%88%E5%99%A8)编码表上的“GB2312”就是指这种表示法。每个汉字及符号以两个[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)来表示。第一个字节称为“高位字节”，第二个字节称为“低位字节”。

“高位字节”使用了0xA1–0xF7（把01–87区的区号加上0xA0），“低位字节”使用了0xA1–0xFE（把01–94加上0xA0）。 由于一级汉字从16区起始，汉字区的“高位字节”的范围是0xB0–0xF7，“低位字节”的范围是0xA1–0xFE，占用的码位是72\*94=6768。其中有5个空位是D7FA–D7FE。

**5.能否相互转换以及兼容**

GBK子集与GBK/GB 18030兼容，GB2312.TXT则不兼容。 后者基于ftp.unicode.org曾经提供的GB2312.TXT实现，[3]于2011年由官方弃用，[4]2016年9月时已无原文件踪迹。此外还有很多种厂商实现。

截至2015年，微软.NET使用的是“GBK子集”实现。ICU[5]、libiconv-1.14、[6]php-5.6、ActivePerl-5.20、Java 1.7、Python 3.4[7]都使用“GB2312.TXT”实现。Ruby 2.2兼容两者编码，但内部使用“GBK子集”实现。W3C的编码技术指南规定，应将gb2312字节流视为GBK编码，与GB18030一并使用同一解码器解码。

**三、BIG5**

**1.来历**

“大五码”（Big5）是由台湾财团法人信息产业策进会为五大中文套装软件所设计的中文共通内码，在1983年12月完成公告，隔年3月，信息产业策进会与台湾13家厂商签定“16位个人电脑套装软件合作开发（BIG-5）项目（五大中文套装软件）”，因为此中文内码是为台湾自行制作开发之“五大中文套装软件”所设计的，所以就称为Big5中文内码。

**2.产生原因**

Big5码的产生，是因为当时个人电脑没有共通的内码，导致厂商推出的中文应用软件无法推广，并且与IBM 5550、王安码等内码，彼此不能兼容；另一方面，台湾当时尚未推出中文编码标准。在这样的时空背景下，为了使台湾早日进入信息时代，所采行的一个项目；同时，这个项目对于以台湾为核心的亚洲繁体汉字圈也产生了久远的影响。

Big5产生前，研发中文电脑的朱邦复认为内码字集应该广纳所有的正异体字，以顾及如户政等应用上的需要，故在当时的内码会议中，建议希望采用他的五万多字的字库。工程师认为虽其技术可行，但是三个字节（超过两个字节）长度的内码却会造成英文显示屏画面映射成中文画面会发生文字无法对齐的问题，因为当时盛行之倚天中文系统画面系以两个字节文字宽度映射成一个中文字图样，英文软件中只要以两个英文字宽度去显示一个中文字，画面就不会乱掉，造成中文系统业者偏爱二个字节长度的内码[9]；此外以仓颉输入码压缩成的内码不具排序等功能，因此未被采用。

在1990年代初期，当中国大陆的电子邮件和转码软件还未普遍之时，在深圳的港商和台商公司亦曾经使用Big5系统，以方便与总部的文件交流、以及避免为大陆的办公室再写一套不同内码的系统。使用简体中文的社区，最常用的是GB 2312、GBK及其后续的国标码（GB 18030）。

除了台湾外，其他使用繁体汉字的地区，如香港（香港增补字符集）、澳门，及使用繁体汉字的海外华人，都曾普遍使用Big5码做为中文内码及交换码。

**3.字节数**

Big5码是一套双字节字符集，使用了双八码存储方法，以两个字节来安放一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。

“高位字节”使用了0x81-0xFE，“低位字节”使用了0x40-0x7E，及0xA1

**4.能否相互转换以及兼容**

大五码的编码码表直接针对存储而设计，每个字符统一使用两个字节存储表示。第1字节范围81H－FEH，避开了同ASCII码的冲突，第2字节范围是40H－7EH和A1H－FEH。因为Big5的字符编码范围同GB2312字符的存储码范围存在冲突，所以在同一正文不能对两种字符集的字符同时支持。

Big5编码的分布如表1－5所示，Big5字符主要部分集中在三个段内：标点符号、希腊字母及特殊符号；常用汉字；非常用汉字。其余部分保留给其他厂商支持。

**四、UTF8**

**1.来历**

UTF-8是一种针对Unicode的可变长度字符编码，也是一种前缀码。它可以用来表示Unicode标准中的任何字符，且其编码中的第一个字节仍与ASCII兼容，这使得原来处理ASCII字符的软件无须或只须做少部分修改，即可继续使用。因此，它逐渐成为电子邮件、网页及其他存储或发送文字的应用中，优先采用的编码。

**2.产生原因**

用ASCII表示的字符使用UNICODE并不高效，因为UNICODE比ASCII占用大一倍的空间，而对ASCII来说高字节的0对他毫无用处。为了解决这个问题，就出现了一些中间格式的字符集，他们被称为通用转换格式，即UTF（Unicode Transformation Format）。常见的UTF格式有：UTF-7, UTF-7.5, UTF-8,UTF-16, 以及 UTF-32。

**3.解决问题**

UTF-8编码规则：如果只有一个字节则其最高二进制位为0；如果是多字节，其第一个字节从最高位开始，连续的二进制位值为1的个数决定了其编码的字节数，其余各字节均以10开头。

表示ASCII字符的UNICODE字符，将会编码成1个字节，并且UTF-8表示与ASCII字符表示是一样的。所有其他的UNICODE字符转化成UTF-8将需要至少2个字节。每个字节由一个换码序列开始。第一个字节由唯一的换码序列，由n位连续的1加一位0组成, 首字节连续的1的个数表示字符编码所需的字节数。

Unicode转换为UTF-8时，可以将Unicode二进制从低位往高位取出二进制数字，每次取6位，如上述的二进制就可以分别取出为如下示例所示的格式，前面按格式填补，不足8位用0填补。

示例

UNICODE uCA(1100 1010) 编码成UTF-8将需要2个字节：

uCA -> C3 8A， 过程如下：

uCA(1100 1010)处于0080 ~07FF之间，从上文中的转换表可知对其编码需要2bytes，即两个字节，其对 应 UTF-8格式为： 110X XXXX10XX XXXX。从此格式中可以看到，对其编码还需要11位，而uCA(1100 1010)仅有8位，这时需要在其二进制数前补0凑成11位: 000 1100 1010, 依次填入110X XXXX 10XX XXXX的空位中， 即得 1100 0011 1000 1010（C38A）。

同理，UNICODE uF03F (1111 0000 0011 1111) 编码成UTF-8将需要3个字节:

u F03F -> EF 80 BF，对应格式为：1110XXXX10XX XXXX10XX XXXX，编码还需要16位，将1111 0000 0011 1111(F03F)依次填入，可得 1110 1111 1000 0000 1011 1111（EF 80 BF）。

**4.字节数**

如果UNICODE字符由2个字节表示，则编码成UTF-8很可能需要3个字节。而如果UNICODE字符由4个字节表示，则编码成UTF-8可能需要6个字节。用4个或6个字节去编码一个UNICODE字符可能太多了，但很少会遇到那样的UNICODE字符。

**5.能否相互转换以及兼容**

修正的UTF-8中，null字符编码成2个字节（1100000010000000）而不是标准的1个字节（00000000），这样作可以保证编码后的字符串中不会嵌入null字符。因此如果在类C语言中处理字符串，文本不会在第一个null字符时截断（C字符串以'\0'结尾）。

在标准UTF-8编码中，超出基本多语言范围（BMP-Basic Multilingual Plane）的字符被编码为4字节格式，但是在修正的UTF-8编码中，他们由代理编码对（surrogatepairs）表示，然后这些代理编码对在序列中分别重新编码。结果标准UTF-8编码中需要4个字节的字符，在修正后的UTF-8编码中将需要6个字节。

**五、UTF16**

**1.来历**

UTF-16是Unicode字符编码五层次模型的第三层：字符编码表（Character Encoding Form，也称为 "storage format"）的一种实现方式。即把Unicode字符集的抽象码位映射为16位长的整数（即码元）的序列，用于数据存储或传递。Unicode字符的码位，需要1个或者2个16位长的码元来表示，因此这是一个变长表示。

**2.产生原因**

用ASCII表示的字符使用UNICODE并不高效，因为UNICODE比ASCII占用大一倍的空间，而对ASCII来说高字节的0对他毫无用处。为了解决这个问题，就出现了一些中间格式的字符集，他们被称为通用转换格式，即UTF（Unicode Transformation Format）。常见的UTF格式有：UTF-7, UTF-7.5, UTF-8,UTF-16, 以及 UTF-32。

**3.解决问题**

UTF-16的大尾序和小尾序储存形式都在用。一般来说，以Macintosh制作或储存的文字使用大尾序格式，以Microsoft或Linux制作或储存的文字使用小尾序格式。

为了弄清楚UTF-16文件的大小尾序，在UTF-16文件的开首，都会放置一个U+FEFF字符作为Byte Order Mark（UTF-16LE以FF FE代表，UTF-16BE以FE FF代表），以显示这个文字档案是以UTF-16编码，其中U+FEFF字符在UNICODE中代表的意义是ZERO WIDTH NO-BREAK SPACE，顾名思义，它是个没有宽度也没有断字的空白。

**4.字节数**

在Unicode基本多文种平面定义的字符（无论是拉丁字母、汉字或其他文字或符号），一律使用2字节储存。而在辅助平面定义的字符，会以代理对（surrogate pair）的形式，以两个2字节的值来储存。

UTF-16比起UTF-8，好处在于大部分字符都以固定长度的字节 (2字节) 储存，但UTF-16却无法兼容于ASCII编码。

**5.能否相互转换以及兼容**

UTF-16可看成是UCS-2的父集。在没有辅助平面字符（surrogate code points）前，UTF-16与UCS-2所指的是同一的意思。但当引入辅助平面字符后，就称为UTF-16了。现在若有软件声称自己支援UCS-2编码，那其实是暗指它不能支援在UTF-16中超过2bytes的字集。对于小于0x10000的UCS码，UTF-16编码就等于UCS码。

**六、UNICODE**

**1.来历**

Unicode字符集编码是Universal Multiple-Octet Coded Character Set 通用多八位编码字符集的简称，是由一个名为 Unicode 学术学会(Unicode Consortium)的机构制订的字符编码系统，支持现今世界各种不同语言的书面文本的交换、处理及显示。该编码于1990年开始研发，1994年正式公布，最新版本是2012年1月31日的Unicode 6.1。Unicode是一种在计算机上使用的字符编码。它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。

**2.编码方式**

Unicode 标准始终使用十六进制数字，而且在书写时在前面加上前缀“U+”，例如字母“A”的编码为 004116 和字符“?”的编码为 20AC16。所以“A”的编码书写为“U+0041”。 unicode的第一个版本是用两个字节(16bit)来表示所有字符.实际上这么说容易让人产生歧义,我们总觉得两个字节就代表保存在计算机中时是两个字节.于是任何字符如果用unicode表示的话保存下来都占两个字节.其实这种说法是错误的.

其实Unicode涉及到两个步骤,首先是定义一个规范,给所有的字符指定一个唯一对应的数字,这完全是数学问题,可以跟计算机没半毛钱关系.第二步才是怎么把字符对应的数字保存在计算机中,这才涉及到实际在计算机中占多少字节空间.

所以我们也可以这样理解,Unicode是用0至65535之间的数字来表示所有字符.其中0至127这128个数字表示的字符仍然跟ASCII完全一样.65536是2的16次方.这是第一步.第二步就是怎么把0至65535这些数字转化成01串保存到计算机中.这肯定就有不同的保存方式了.于是出现了UTF(unicode transformation format),有UTF-8,UTF-16.

**3.能否相互转换以及兼容**

通用字符集是与UNICODE同类的组织，UCS-2和UNICODE兼容。 位数：它有UCS-2和UCS-4两种格式，分别是2字节和4字节。 范围：UCS-4只是在UCS-2前面加了0×0000。

UNICODE字符集

作用：为世界650种语言进行统一编码，兼容ISO-8859-1。

位数：UNICODE字符集有多个编码方式，分别是UTF-8，UTF-16和UTF-32。

**五、总结**

**本次小课题研究主要针对各种字符集，大体上是让我们从这些编码集的基础掌握。去学习以及掌握这些编码集的实现方式和兼容问题。**

GB2312编码  
当天朝人们得到计算机时，规定：一个小于127的字符的意义与原来相同，但两个大于127的字符连在一起时，就表示一个汉字，前面的一个字节（他称之为高字节）从0xA1用到0xF7，后面一个字节（低字节）从0xA1到0xFE，这样我们就可以组合出大约7000多个简体汉字了。在这些编码里，我们还把数学符号、罗马希腊的字母、日文的假名们都编进去了，连在ASCII里本来就有的数字、标点、字母都统统重新编了两个字节长的编码，这就是常说的“全角”字符，而原来在127号以下的那些就叫"半角"字符了。于是就把这种汉字方案叫做“GB2312”。GB2312是对ASCII的中文扩展。一个汉字两个字节即一个汉字两个英文字符由此而来。  
  
  
  
  
UNICODE编码  
ISO （国际标谁化组织）的国际组织决定着手统一字符集。他们采用的方法很简单：废了所有的地区性编码方案，重新搞一个包括了地球上所有文化、所有字母和符号的编码！他们打算叫它 UCS, 俗称 UNICODE 。

（ Universal Multiple-Octet Coded Character Set ）  
在UNICODE中，一个汉字算两个英文字符的时代已经快过去了。无论是半角的英文字母，还是全角的汉字，它们都是统一的“一个字符”！同时，也都是统一的“两个字节”。  
  
  
六、UTF-8和UTF-16  
UNICODE 如何在网络上传输也是一个必须考虑的问题，于是面向传输的众多 UTF（UCSTransferFormat）标准出现了，顾名思义，UTF8就是每次8个位传输数据，而UTF16就是每次16个位，只不过为了传输时的可靠性，从UNICODE到UTF时并不是直接的对应，而是要过一些算法和规则来转换。

**参考文献**：

①维基百科 <https://en.wikipedia.org/wiki/Wiki>。

②CSDN论坛 <http://bbs.csdn.net/topics/100037062>。

③百度百科 <https://baike.baidu.com/>。