关于常见编码的认识

姓名：付俊怡

学号：201503469

单位：软工11502班

指导老师：向华

**关于常见编码的认识**

摘要：在[计算机硬件](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A1%AC%E4%BB%B6" \t "https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/_blank)中，[编码](https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/80092" \t "https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/_blank)是指用[代码](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81" \t "https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/_blank)来表示各组数据资料，使其成为可利用计算机进行处理和分析的信息。代码是用来表示事物的记号，它可以用数字、字母、特殊的符号或它们之间的组合来表示将[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE" \t "https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/_blank)转换为代码或编码字符，并能译为原数据形式。是计算机书写指令的过程，[程序设计](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1" \t "https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/_blank)中的一部分。在地图自动制图中，按一定规则用数字与字母表示地图内容的过程，通过编码，使计算机能识别地图的各地理要素。

关键字：GB2312，Big5，Unicode，UTF-8/UTF-16

目录

一 序言.............................................................3

二 GB2312...........................................................3

2.1产生的原因......................................................3

2.2解决的问题......................................................3

2.3字节数..........................................................3

2.4描述方式........................................................4

三Big5.............................................................4

3.1产生的原因........................................................4

3.2解决的问题........................................................4

3.3字节数..........................................................4

3.4描述方式........................................................4

四Unicode

4.1 产生的原因......................................................5

4.2 解决的问题......................................................5

4.3字节数..........................................................5

4.4描述方式........................................................5

五UTF-8............................................................6

5.1产生的原因......................................................6

5.2 解决的问题......................................................6

5.3字节数..........................................................7

5.4描述方式........................................................7

六UTF-16...........................................................7

6.1产生的原因......................................................7

6.2解决的问题......................................................7

6.3字节数..........................................................8

6.4描述方式........................................................8

七 参考文献.........................................................8

一、序言

由于计算机是美国人发明的，因此，最早只有127个字母被编码到计算机里，也就是大小写英文字母、数字和一些符号，这个编码表被称为ASCII编码，比如大写字母A的编码是65，小写字母z的编码是122。但是要处理中文显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。你可以想得到的是，全世界有上百种语言，日本把日文编到Shift\_JIS里，韩国把韩文编到Euc-kr里，各国有各国的标准，就会不可避免地出现冲突，结果就是，在多语言混合的文本中，显示出来会有乱码。因此，Unicode应运而生。Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。

新的问题又出现了：如果统一成Unicode编码，乱码问题从此消失了。但是，如果你写的文本基本上全部是英文的话，用Unicode编码比ASCII编码需要多一倍的存储空间，在存储和传输上就十分不划算。

所以，本着节约的精神，又出现了把Unicode编码转化为“可变长编码”的UTF-8编码。UTF-8编码把一个Unicode字符根据不同的数字大小编码成1-6个字节，常用的英文字母被编码成1个字节，汉字通常是3个字节，只有很生僻的字符才会被编码成4-6个字节。如果你要传输的文本包含大量英文字符，用UTF-8编码就能节省空间。UTF-8编码有一个额外的好处，就是ASCII编码实际上可以被看成是UTF-8编码的一部分，所以，大量只支持ASCII编码的历史遗留软件可以在UTF-8编码下继续工作。

1. GB2312

2.1产生的原因

《信息交换用汉字编码字符集》是由中国国家标准总局1980年发布，1981年5月1日开始实施的一套国家标准，标准号是[GB 2312](https://baike.baidu.com/item/GB 2312" \t "https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E4%BA%A4%E6%8D%A2%E7%94%A8%E6%B1%89%E5%AD%97%E7%BC%96%E7%A0%81%E5%AD%97%E7%AC%A6%E9%9B%86/_blank)—1980。GB2312编码适用于汉字处理、汉字通信等系统之间的信息交换，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。基本集共收入汉字6763个和非汉字图形字符682个。整个字符集分成94个区，每区有94个位。每个区位上只有一个字符，因此可用所在的区和位来对汉字进行编码，称为[区位码](https://baike.baidu.com/item/%E5%8C%BA%E4%BD%8D%E7%A0%81" \t "https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E4%BA%A4%E6%8D%A2%E7%94%A8%E6%B1%89%E5%AD%97%E7%BC%96%E7%A0%81%E5%AD%97%E7%AC%A6%E9%9B%86/_blank)。把换算成十六进制的区位码加上2020H，就得到[国标码](https://baike.baidu.com/item/%E5%9B%BD%E6%A0%87%E7%A0%81" \t "https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E4%BA%A4%E6%8D%A2%E7%94%A8%E6%B1%89%E5%AD%97%E7%BC%96%E7%A0%81%E5%AD%97%E7%AC%A6%E9%9B%86/_blank)。国标码加上8080H，就得到常用的计算机机内码。1995年又颁布了《汉字编码扩展规范》（GBK）。GBK与GB 2312—1980国家标准所对应的内码标准兼容，同时在字汇一级支持ISO/IEC10646—1和GB 13000—1的全部中、日、韩（CJK）汉字，共计20902字。

2.2解决的问题

GB2312是中国国家标准的简体中文字符集。它所收录的汉字已经覆盖99.75%的使用频率，基本满足了汉字的计算机处理需要。在中国大陆和新加坡获广泛使用。

GB2312收录简化汉字及一般符号、序号、数字、拉丁字母、日文假名、希腊字母、俄文字母、汉语拼音符号、汉语注音字母，共7445个图形字符。其中包括6763个汉字，其中一级汉字3755个，二级汉字3008个；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个全角字符。

2.3字节数

在使用GB 2312的程序通常采用EUC储存方法，以便兼容于ASCII。这种格式称为EUC-CN。浏览器编码表上的“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。例如“啊”字在大多数程序中，会以两个字节，0xB0（第一个字节）0xA1（第二个字节）储存。（与区位码对比：0xB0=0xA0+16,0xA1=0xA0+1）。

2.4描述方式

（1）分区表示：

GB2312中对所收汉字进行了“分区”处理，每区含有94个汉字/符号。这种表示方式也称为区位码。各区包含的字符如下：01-09区为特殊符号；16-55区为一级汉字，按拼音排序；56-87区为二级汉字，按部首/笔画排序；10-15区及88-94区则未有编码。

（2）双字节表示

两个字节中前面的字节为第一字节，后面的字节为第二字节。习惯上称第一字节为“高字节”，而称第二字节为“低字节”。“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)。

1. Big5

3.1产生的原因

Big5码的产生，是因为当时台湾不同厂商各自推出不同的编码，如IBM 5550、王安码等，彼此不能兼容；另一方面，台湾当时尚未推出官方的汉字编码，而中国内地所推行的GB 2312编码，亦未有收录繁体字。同时，这个计划对于以台湾为核心的亚洲汉字圈也产生了久远的影响。

大五码是由资策会于1984年策划制定，拥有13053个中文字、408个字符以及33个控制字元的字集，是我国早期中文电脑的业界标准，也是中文社群最常用的电子汉字字集标准。而后随着电脑扩充需要，业界各操作系统开发商推出了不同版本的大五码，为统一标准，经济部标准检验局在2003年委托财团法人中国数位化技术推广基金会修改了大五码编码字元表，重整为Big5-2003版本。

3.2解决的问题

Big5字符集共收录13,053个中文字，该字符集在中国台湾使用。耐人寻味的是该字符集重复地收录了两个相同的字：“兀”(0xA461及0xC94A)、“嗀”(0xDCD1及0xDDFC)。

3.3字节数

Big5码使用了双字节储存方法，以两个字节来编码一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。高位字节的编码范围0xA1-0xF9，低位字节的编码范围0x40-0x7E及0xA1-0xFE。

各编码范围对应的字符类型如下：0xA140-0xA3BF为标点符号、希腊字母及特殊符号，另外于0xA259-0xA261，存放了双音节度量衡单位用字：兙兛兞兝兡兣嗧瓩糎；0xA440-0xC67E为常用汉字，先按笔划再按部首排序；0xC940-0xF9D5为次常用汉字，亦是先按笔划再按部首排序。

3.4描述方式

在Big5的分区中：0x8140-0xA0FE保留给用户自定义字符（造字区） 0xA140-0xA3BF 标点符号、希腊字母及特殊符号，包括在0xA259-0xA261，安放了九个计量用汉字：兙兛兞兝兡兣嗧瓩糎。0xA3C0-0xA3FE 保留。此区没有开放作造字区用。0xA440-0xC67E 常用汉字，先按笔划再按部首排序。 0xC6A1-0xC8FE 保留给用户自定义字符（造字区0xC940-0xF9D5次常用汉字，亦是先按笔划再按部首排序。 0xF9D6-0xFEFE 保留给用户自定义字符（造字区）值得留意的是，Big5重复收录了两个相同的字：“兀、兀”（0xA461[U+5140]及0xC94A[U+FA0C]）、“嗀、嗀”（0xDCD1[U+55C0]及0xDDFC[U+FA0D]）。此外“十”、“卅”也在符号区又重复了一次，在检索系统中常会造成查询不到字。

1. Unicode

4.1产生的原因

Unicode 是为了解决传统的字符编码方案的局限而产生的，例如ISO 8859所定义的字符虽然在不同的国家中广泛地使用，可是在不同国家间却经常出现不兼容的情况。很多传统的编码方式都有一个共同的问题，即容许电脑处理双语环境（通常使用拉丁字母以及其本地语言），但却无法同时支持多语言环境（指可同时处理多种语言混合的情况）。

Unicode 编码包含了不同写法的字，如“ɑ/a”、“户/户/戸”。然而在汉字方面引起了一字多形的认定争议。

在文字处理方面，统一码为每一个字符而非字形定义唯一的代码（即一个整数）。换句话说，统一码以一种抽象的方式（即数字）来处理字符，并将视觉上的演绎工作（例如字体大小、外观形状、字体形态、文体等）留给其他软件来处理，例如网页浏览器或是文字处理器。几乎所有电脑系统都支持基本拉丁字母，并各自支持不同的其他编码方式。Unicode为了和它们相互兼容，其首256字符保留给ISO 8859-1所定义的字符，使既有的西欧语系文字的转换不需特别考量；并且把大量相同的字符重复编到不同的字符码中去，使得旧有纷杂的编码方式得以和Unicode编码间互相直接转换，而不会丢失任何信息。举例来说，全角格式区段包含了主要的拉丁字母的全角格式，在中文、日文、以及韩文字形当中，这些字符以全角的方式来呈现，而不以常见的半角形式显示，这对竖排文字和等宽排列文字有重要作用。

在表示一个Unicode的字符时，通常会用“U+”然后紧接着一组十六进制的数字来表示这一个字符。在基本多文种平面（英文为 Basic Multilingual Plane，简写 BMP。它又简称为“零号平面”, plane 0）里的所有字符，要用四位十六进制数（例如U+4AE0，共支持六万多个字符）；在零号平面以外的字符则需要使用五位或六位十六进制数了。旧版的Unicode标准使用相近的标记方法，但却有些微的差异：在Unicode 3.0里使用“U-”然后紧接着八位数，而“U+”则必须随后紧接着四位数。

4.2解决的问题

能够使计算机实现跨语言、跨平台的文本转换及处理。

4.3字节数

Unicode编码长度可达到32位，即4字节。第一字节称为组，第二字节称为面，第三字节称为行，第四字节称为点。第0组第0面里的字符可以只用2个字节表示，且涵盖了绝大部分的常用字，所以备受青睐。为了方便称呼，Unicode给它了一个名称——基本多文种平面（BMP）。基本多文种平面值域和上域都是0到FFFF，共计65535个码点。8 9基本多文种平面是Unicode的子集。

虽然Unicode的表示长度是32位，但因为通常表示的是常用字，且常用字在BMP的字库里，所以表示长度往往缩减为16位。

4.4描述方式

Unicode是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。目前的Unicode字符分为17组编排，0x0000 至 0xFFFF，每组称为平面（Plane），而每平面拥有65536个码位，共1114112个。然而目前只用了少数平面。UTF-8、UTF-16、UTF-32都是将数字转换到程序数据的编码方案。

通用字符集（Universal Character Set, UCS）是由ISO制定的ISO 10646（或称ISO/IEC 10646）标准所定义的标准字符集。UCS-2用两个字节编码，UCS-4用4个字节编码。

历史上存在两个独立的尝试创立单一字符集的组织，即国际标准化组织（ISO）和多语言软件制造商组成的统一码联盟。前者开发的 ISO/IEC 10646 项目，后者开发的统一码项目。因此最初制定了不同的标准。

1991年前后，两个项目的参与者都认识到，世界不需要两个不兼容的字符集。于是，它们开始合并双方的工作成果，并为创立一个单一编码表而协同工作。从Unicode 2.0开始，Unicode采用了与ISO 10646-1相同的字库和字码；ISO也承诺，ISO 10646将不会替超出U+10FFFF的UCS-4编码赋值，以使得两者保持一致。两个项目仍都存在，并独立地公布各自的标准。但统一码联盟和ISO/IEC JTC1/SC2都同意保持两者标准的码表兼容，并紧密地共同调整任何未来的扩展。在发布的时候，Unicode一般都会采用有关字码最常见的字型，但ISO 10646一般都尽可能采用Century字型。

UCS-4根据最高位为0的最高字节分成27=128个group。每个group再根据次高字节分为256个平面（plane）。每个平面根据第3个字节分为256行 （row），每行有256个码位（cell）。group 0的平面0被称作BMP（Basic Multilingual Plane）。如果UCS-4的前两个字节为全零，那么将UCS-4的BMP去掉前面的两个零字节就得到了UCS-2。每个平面有216=65536个码位。Unicode计划使用了17个平面，一共有17×65536=1114112个码位。在Unicode 5.0.0版本中，已定义的码位只有238605个，分布在平面0、平面1、平面2、平面14、平面15、平面16。其中平面15和平面16上只是定义了两个各占65534个码位的专用区（Private Use Area），分别是0xF0000-0xFFFFD和0x100000-0x10FFFD。所谓专用区，就是保留给大家放自定义字符的区域，可以简写为PUA。

平面0也有一个专用区：0xE000-0xF8FF，有6400个码位。平面0的0xD800-0xDFFF，共2048个码位，是一个被称作代理区（Surrogate）的特殊区域。代理区的目的用两个UTF-16字符表示BMP以外的字符。在介绍UTF-16编码时会介绍。

如前所述在Unicode 5.0.0版本中，238605-65534\*2-6400-2048=99089。余下的99089个已定义码位分布在平面0、平面1、平面2和平面14上，它们对应着Unicode定义的99089个字符，其中包括71226个汉字。平面0、平面1、平面2和平面14上分别定义了52080、3419、43253和337个字符。平面2的43253个字符都是汉字。平面0上定义了27973个汉字。

在Unicode中：汉字“字”对应的数字是23383（十进制），十六进制表示为5B57。在Unicode中，我们有很多方式将数字23383表示成程序中的数据，包括：UTF-8、UTF-16、UTF-32。UTF是“Unicode Transformation Format”的缩写，可以翻译成Unicode字符集转换格式，即怎样将Unicode定义的数字转换成程序数据。

这里用char、char16\_t、char32\_t分别表示无符号8位整数，无符号16位整数和无符号32位整数。UTF-8、UTF-16、UTF-32分别以char、char16\_t、char32\_t作为编码单位。（注： char16\_t 和 char32\_t 是 C++ 11 标准新增的关键字。如果你的编译器不支持 C++ 11 标准，请改用 unsigned short 和 unsigned long。）“汉字”的UTF-8编码需要6个字节。“汉字”的UTF-16编码需要两个char16\_t，大小是4个字节。“汉字”的UTF-32编码需要两个char32\_t，大小是8个字节。根据字节序的不同，UTF-16可以被实现为UTF-16LE或UTF-16BE，UTF-32可以被实现为UTF-32LE或UTF-32BE。下面介绍UTF-8、UTF-16、UTF-32、字节序和BOM。

1. UTF-8

5.1产生的原因

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。

5.2解决的问题

UTF-8便于不同的计算机之间使用网络传输不同语言和编码的文字，使得双字节的Unicode能够在现存的处理单字节的系统上正确传输。

5.3字节数

UTF-8使用可变长度字节来储存 Unicode字符，例如ASCII字母继续使用1字节储存，重音文字、希腊字母或西里尔字母等使用2字节来储存，而常用的汉字就要使用3字节。辅助平面字符则使用4字节。

UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，又称万国码。由Ken Thompson于1992年创建。现在已经标准化为RFC 3629。UTF-8用1到6个字节编码UNICODE字符。用在网页上可以同一页面显示中文简体繁体及其它语言（如英文，日文，韩文）。

5.4描述方式

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）。

1. UTF-16

6.1产生的原因

UTF-16是Unicode的其中一个使用方式。 UTF是 Unicode TransferFormat，即把Unicode转做某种格式的意思。

它定义于ISO/IEC 10646-1的附录Q，而RFC2781也定义了相似的做法。

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。

6.2解决的问题

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。

6.3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。

6.4描述方式

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。

1. 参考文献
2. 维基百科 <https://en.wikipedia.org/wiki/Wiki>。
3. 百度百科 <https://baike.baidu.com/>。