

# 字符集及字符编码问题

## 一、字符集及字符编码概述

### (一)、字符编码的重要性

只要有中文的地方就会出现编码问题。

编码问题是学习编程中非常重要的一个问题，如果对编码和字符集不能透彻理解，那么就如同行走在沼泽，随时会让程序员深陷其中、无法自拔。字符编码这个知识如此重要，但现实是，很多公司里即便从事多年开发的程序员也常常对此一知半解。遇到问题时大多连蒙带猜地解决，不求甚解。

### (二)、工作中最常出现编码问题的地方

- 1、无论前端开发、java开发、PHP开发、移动开发还是Python开发，任何一种计算机语言都会遇到字符编码问题；
- 2、表单提交数据到服务器时，容易出现数据传输错误；
- 3、只要涉及到网络传输中文数据，极其出现编码问题；
- 4、当用到Ajax技术时，经常出现乱码问题；
- 5、非编程人员日常生活中也常出现编码问题。比如打开文件后显示乱码，如果理解字符编码，只需要换一种打开方式，或许问题就迎刃而解。

总之，**字符编码问题很重要，不掌握将后患无穷。**

### (三)、基本概念

#### 1、首先介绍：字符、字符集、字符编码三者的含义。

(1)、**字符**：是关于文字和符号的总称，包括各个国家的文字、标点符号、图形符号、数字等等。例如：一个汉字，一个标点符号逗号，一个英文字母，一个数字，这都是字符。

(2)、**字符集**：是多个字符的集合。我们可以理解成就是一本大字典。字符集种类很多，每个字符集包含的字符个数也不同，常见的字符集有：ASCII字符集，Unicode字符集，GB2312字符集 ISO-8859-1字符集等等。

(3)、**字符编码**：计算机只能识别二进制1和0。字符集中的字符，计算机是不能直接识别的，所以要将字符集转化为计算机可以识别的二进制，这个转化过程就是编码，而字符编码就是将二进制的数与字符集中的字符对应起来的一套规则。

各个国家和地区在制定编码标准的时候，字符集和编码规则都是同时制定的。因此，平常我们所说的字符集，如ASCII、GB2312、GBK、Unicode等，除了有字符集合的含义外，同时也包含编码的含义。

一般来说，每种字符集都有相应的一种字符编码规则，如ASCII码、GB2312字符集等都只有一种编码方式。但有的字符集有多种编码方式，比如，Unicode字符集有UTF-8、UTF-16、UTF-32等多种字符编码。

#### 2、其次介绍**字符**与**字节**的含义。

不要将字符与字节搞混。字符是文化符号，而字节是文件的长度单位。

比如有一个文件，内容如下：

**“ABC123”**

在这个文件中，我们输入的是“半角”的“ABC123”，一共包含6个字符。但是这个文件有多大，占几个字节呢？



答案是：6字节、9字节、12字节、14字节、24字节、28字节

怎么会有这么多答案呢？其实只要掌握了字符集及字符编码就明白了。不同的字符集、不同的编码方式，都会导致文件的大小不同。

### (四)、回顾计算机基本原理

1. 计算机最基本的硬件是**半导体芯片**，每块芯片集成了**数万到数百万个晶体管**；
2. 计算机最基本的物理元件是晶体管，每一个晶体管具有开和关两种状态；
3. 七个晶体管组合在一起就可以组合出**128种**不同的状态(=2的7次方)；

4. 八个晶体管组合在一起就可以组合出256种不同的状态(=2的8次方)。

									
一张扑克	0	1							
两张扑克	00	11	01	10					
三张扑克	011	001	000	100	110	111	010	101	

二、ASCII码

(一)、ASCII码的由来

- 1. ASCII码 (American Standard Code for Information Interchange) ，它是“**美国标准信息互换码**”；
  - ASCII码是由**美国国家标准学会**(American National Standard Institute , ANSI )制定的，标准的单字节字符编码方案，用于基于文本的数据。起始于50年代后期，在**1967**年定案。它最初是美国国家标准，供不同计算机在相互通信时用作共同遵守的西文字符编码标准，它已被国际标准化组织 (International Organization for Standardization, ISO) 定为国际标准，称为ISO 646标准。适用于所有拉丁文字字母。
  - 美国国家标准学会 (American National Standards Institute: ANSI) 。美国国家标准学会成立于1918年。当时，美国的许多企业和专业技术团体，已开始了标准化工作，但因彼此间没有协调，存在不少矛盾和问题。为了进一步提高效率，数百个科技学会、协会组织和团体，均认为有必要成立一个专门的标准化机构，并制订统一的通用标准。
- 2. 计算机每个晶体管可以存储的两种状态，构成了计算机最小的存储单位：位 (bit，比特) ；
- 3. 七个晶体管组合出的128种不同的状态，每种状态保存一个字节，这就构成了“**标准的7位ASCII码**”；
  - 128种状态足够保存美国所有的文化符号（大写26个字母、小写26个字母、0-9阿拉伯数字、所有的英文标点符号、数学运算符号、其他特殊符号以及控制码）；
  - ASCII码中编号0~**31**是**非打印字符**，用于控制字符的换行回车删除等。也统称为“**控制码**”；32~126 是打印字符，可以通过键盘输入并且能够显示出来。
  - 编号**48**至57，保存的是数字0到9；
  - 编号**65**至90，保存的是大写字母A到Z；
  - 编号**97**至122，保存的是小写字母a到z；
  - 其它位置保存的是标点符号、数学运算符号及特殊符号。
- 4. 八个晶体管可以组合256种不同的状态，这就是“**扩展ASCII码**”；
  - 计算机起初是美国用于军事而发明的产物，后来更多国家开始使用计算机，这些国家的字母是ASCII码中没有的；
  - 后来**IBM公司**在此基础上进行了扩展，用8bit来表示一个字符，总共可以表示256个字符。当第一位是0时仍然表示7位ASCII码的字符，当第一位为1时就表示新补充的字符。
  - 编号128到编号255的字符集被称为“**扩展ASCII码**”。

(二)、ASCII码的编号及对应字符

二进制	十进制	十六进制	缩写	解释	二进制	十进制	十六进制	字符	二进制	十进制	十六进制	字符	二进制	十进制	十六进制	字符
0000 0000	0	0	NUL	空字符(Null)	0010 0000	32	20	空格	1	65	41	A	0110 0001	97	61	a
0000 0001	1	1	SOH	标题开始	0010 0001	33	21	!	0	66	42	B	0110 0010	98	62	b
0000 0010	2	2	STX	正文开始	0010 0010	34	22	"	1	67	43	C	0110 0011	99	63	c
0000 0011	3	3	ETX	正文结束	0010 0011	35	23	#	0	68	44	D	0110 0100	100	64	d
0000 0100	4	4	EOT	传输结束	0010 0100	36	24	\$	1	69	45	E	0110 0101	101	65	e
	5	5	ENQ	请求	0010 0101	37	25	%	0	70	46	F	0110 0110	102	66	f
0000 0110	6	6	ACK	收到通知	0010 0110	38	26	&	1	71	47	G	0110 0111	103	67	g
0000 0111	7	7	BEL	响铃	0010 0111	39	27	'	0	72	48	H	0110 1000	104	68	h
0000 1000	8	8	BS	退格	0010 1000	40	28	(	1	73	49	I	0110 1001	105	69	i
0000 1001	9	9	HT	水平制表符	0010 1001	41	29	)	0	74	4A	J	0110 1010	106	6A	j
0000 1010	10	0A	LF	换行键	0010 1010	42	2A	*	1	75	4B	K	0110 1011	107	6B	k
0000 1011	11	0B	VT	垂直制表符	0010 1011	43	2B	+	0	76	4C	L	0110 1100	108	6C	l
0000 1100	12	0C	FF	换页键	0010 1100	44	2C	,	1	77	4D	M	0110 1101	109	6D	m
0000 1101	13	0D	CR	回车键	0010 1101	45	2D	-	0	78	4E	N	0110 1110	110	6E	n
0000 1110	14	0E	SO	不用切换	0010 1110	46	2E	.	1	79	4F	O	0110 1111	111	6F	o
0000 1111	15	0F	SI	启用切换	0010 1111	47	2F	/	0	80	50	P	0111 0000	112	70	p
0001 0000	16	10	DLE	数据链路转	0011 0000	48	30	0	1	81	51	Q	0111 0001	113	71	q
0001 0001	17	11	DC1	设备控制1	0011 0001	49	31	1	0	82	52	R	0111 0010	114	72	r
0001 0010	18	12	DC2	设备控制2	0011 0010	50	32	2	1	83	53	S	0111 0011	115	73	s
0001 0011	19	13	DC3	设备控制3	0011 0011	51	33	3	0	84	54	T	0111 0100	116	74	t
0001 0100	20	14	DC4	设备控制4	0011 0100	52	34	4	1	85	55	U	0111 0101	117	75	u
0001 0101	21	15	NAK	拒绝接收	0011 0101	53	35	5	0	86	56	V	0111 0110	118	76	v
0001 0110	22	16	SYN	同步空闲	0011 0110	54	36	6	1	87	57	W	0111 0111	119	77	w
0001 0111	23	17	ETB	传输块结束	0011 0111	55	37	7	0	88	58	X	0111 1000	120	78	x
0001 1000	24	18	CAN	取消	0011 1000	56	38	8	1	89	59	Y	0111 1001	121	79	y
0001 1001	25	19	EM	介质中断	0011 1001	57	39	9	0	90	5A	Z	0111 1010	122	7A	z
0001 1010	26	1A	SUB	替补	0011 1010	58	3A	:	1	91	5B	[	0111 1011	123	7B	{
0001 1011	27	1B	ESC	溢出	0011 1011	59	3B	;	0	92	5C	\	0111 1100	124	7C	
0001 1100	28	1C	FS	文件分隔符	0011 1100	60	3C	<	1	93	5D	]	0111 1101	125	7D	}
0001 1101	29	1D	GS	分组符	0011 1101	61	3D	=	0	94	5E	^	0111 1110	126	7E	~
0001 1110	30	1E	RS	记录分隔符	0011 1110	62	3E	>	1	95	5F	_				
0001 1111	31	1F	US	单元分隔符	0011 1111	63	3F	?	0	96	60	`				
0111 1111	127	7F	DEL	删除	0100 0000	64	40	@								

ASCII 字符代码表 一

高四位		ASCII 非打印控制字符										ASCII 打印字符													
		0000					0001					0010		0011		0100		0101		0110		0111			
		0					1					2		3		4		5		6		7			
低四位		+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	ctrl	
0000		0	0		BLANK NULL	^@	NUL	空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p
0001		1	1	☺	^A	SOH	标题开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010		2	2	☹	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011		3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100		4	4	♦	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t	
0101		5	5	♣	^E	ENQ	查询	21	§	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u	
0110		6	6	♠	^F	ACK	确认	22	■	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111		7	7	●	^G	BEL	震铃	23	↑	^W	ETB	传输块结束	39	'	55	7	71	G	87	w	103	g	119	w	
1000		8	8	◻	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001		9	9	◯	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010		A	10	◼	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z	
1011		B	11	♂	^K	VT	垂直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{	
1100		C	12	♀	^L	FF	换页/新页	28	└	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124		
1101		D	13	🎵	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110		E	14	🎵	^N	SO	移出	30	▲	^_	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111		F	15	☼	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	Back space

注：表中的ASCII字符可以用:ALT + “小键盘上的数字键” 输入



ASCII 字符代码表 二

高四位 低四位		扩充ASCII码字符集															
		1000		1001		1010		1011		1100		1101		1110		1111	
		8		9		A/10		B/16		C/32		D/48		E/64		F/80	
		+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符
0000	0	128	Ç	144	É	160	á	176	⌘	192	⌘	208	⌘	224	α	240	≡
0001	1	129	ü	145	æ	161	í	177	⌘	193	⌘	209	⌘	225	β	241	±
0010	2	130	é	146	Æ	162	ó	178	⌘	194	⌘	210	⌘	226	Γ	242	≥
0011	3	131	â	147	ô	163	ú	179	⌘	195	⌘	211	⌘	227	π	243	≤
0100	4	132	ä	148	ö	164	ñ	180	⌘	196	⌘	212	Ô	228	Σ	244	∫
0101	5	133	à	149	ò	165	Ñ	181	⌘	197	⌘	213	⌘	229	σ	245	∫
0110	6	134	å	150	û	166	ª	182	⌘	198	⌘	214	⌘	230	μ	246	÷
0111	7	135	ç	151	ù	167	º	183	⌘	199	⌘	215	⌘	231	τ	247	≈
1000	8	136	ê	152	ÿ	168	¿	184	⌘	200	⌘	216	⌘	232	Φ	248	°
1001	9	137	ë	153	Ö	169	⌘	185	⌘	201	⌘	217	⌘	233	Θ	249	•
1010	A	138	è	154	Ü	170	⌘	186	⌘	202	⌘	218	⌘	234	Ω	250	•
1011	B	139	ï	155	¢	171	½	187	⌘	203	⌘	219	⌘	235	δ	251	√
1100	C	140	î	156	£	172	¼	188	⌘	204	⌘	220	⌘	236	∞	252	∞
1101	D	141	ì	157	¥	173	¡	189	⌘	205	⌘	221	⌘	237	φ	253	²
1110	E	142	Ä	158	⌘	174	«	190	⌘	206	⌘	222	⌘	238	ε	254	■
1111	F	143	Å	159	ƒ	175	»	191	⌘	207	⌘	223	⌘	239	∩	255	BLANK FF

注：表中的ASCII字符可以用：ALT + “小键盘上的数字键” 输入

### 三、中文字符集

#### (一)、GB2312字符集

1. 中国人们得到计算机时，已经没有可以利用的字节状态来表示汉字，况且中文常用字就有6000多个；
2. GB2312字符集中依然保留了前128个ASCII码的字符。而把编号在127号之后的符号们全部去掉。
  - 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，这就是“GB2312字符集”，这是由中国国家标准总局1980年发布的；
  - GB2312是对ASCII编码的中文扩展，ASCII码是典型的单字节编码，一个字符就占一个字节长度。因此GB2312字符集中，前128个字符是单字节的，编号在127之后的GB2312编码则是双字节的，也就是一个字符是两个字节长度。
  - GB2312字符集只支持6763个简体汉字。
3. GB2312字符集，除汉字外还把数学符号、罗马字母、希腊字母、日文假名都编进去了。此外还将ASCII里本来就有的大小写英文字母、阿拉伯数字、标点符号重新编了一套两个字节长的编码，这就是“全角”字符，而原来在127号以下的那些就叫“半角”字符了。全角和半角只针对于字母、数字和标点而言的，对于中文字是无所谓全角和半角区分的。
  - “全角”字符是双字节编码，而“半角”字符是单字节编码，因此长度上不同；例如：
  - a b c d e f g A B C D E F G 1 2 3 4 5 6 7 8 9 0
  - abcdefgABCDEFG1234567890

## GB2312简体中文编码表

code	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A1A0																
A1B0	“	”	(	)	<	>	《	》	「	」	『	』	【	】		
A1C0	±	×	÷	:	∧	∨	Σ	Π	U	∩	∈	::	√	⊥	//	∠
A1D0	ˆ	˚	∫	∫	≡	≈	∞	∞	≠	≠	≠	≠	≠	≠	∞	∞
A1E0	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴
A1F0	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	=	

code	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A2A0	i	ii	iii	iv	v	vi	vii	viii	ix	x						
A2B0	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	
A2C0	16.	17.	18.	19.	20.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
A2D0	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)	(25)	(26)	(27)
A2E0	(28)	(29)	(30)	(31)	(32)	(33)	(34)	(35)	(36)	(37)	(38)	(39)	(40)	(41)	(42)	(43)
A2F0	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII				

code	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A3A0	!	!"	#	¥	%	&	'	(	)	*	+	,	-	.	/	
A3B0	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
A3C0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A3D0	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
A3E0	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
A3F0	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

code	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A4A0	あ	あ	い	い	う	う	え	え	お	お	か	が	き	ぎ	く	
A4B0	ぐ	け	こ	こ	さ	さ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	
A4C0	だ	ち	づ	づ	て	て	と	ど	な	に	ぬ	ね	の	は		
A4D0	ば	ば	ひ	ひ	ふ	ふ	へ	べ	ほ	ぼ	ま	み	め	も	わ	
A4E0	む	め	も	や	や	ゆ	ゆ	よ	ら	り	る	れ	ろ	わ		
A4F0	ゑ	ゑ	ん													

code	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A5A0	ア	ア	イ	イ	ウ	ウ	エ	エ	オ	オ	カ	ガ	キ	ギ	ク	
A5B0	グ	ケ	コ	コ	サ	サ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	
A5C0	チ	チ	ツ	ツ	テ	テ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ		
A5D0	パ	パ	ヒ	ヒ	フ	フ	ヘ	ベ	ホ	ボ	マ	ミ	メ	モ	ワ	
A5E0	ム	メ	モ	ヤ	ヤ	ユ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ		
A5F0	カ	エ	ラ	ン	ヴ	カ	ケ									

### (二)、GBK 字符集

中国的汉字太多，GB2312字符集中的6763个汉字不够使用，1993年又颁布了《汉字编码扩展规范》，也就是GBK 编码，GBK 包括 GB2312 的所有内容，同时又增加了近20000个新的汉字（包括繁体字）和符号。

### (三)、GB18030字符集

随着中国各少数民族使用电脑，中文字符集继续扩展。国家标准GB18030-2000《信息交换用汉字编码字符集基本集的补充》是我国继GB2312-1980和GB13000-1993之后最重要的汉字编码标准，GB18030字符集又加了几千个新的少数民族的文字。GB18030-2000是GBK的取代版本，它的主要特点是在GBK基础上增加了CJK（china-japan-korea，中日韩）统一汉字扩充的汉字。

### (四)、DBCS字符集

GB2312、GBK、GB18030等一系列汉字编码，中国程序员通称他们叫做"DBCS"（Double Byte Charecter Set 双字节字符集）。其最大的特点是：除ASCII码字符外，其它的每个字符都占两个字节长度。

## 四、Unicode字符集（UCS，统一字符集、万国码、单一码）

### (一)、出现的原因

1、当时各个国家都像中国这样搞出一套自己的编码标准，结果相互之间谁也不懂谁的编码，谁也不支持别人的编码，甚至中国大陆和中国台湾这样只相隔了150海里，使用着同一种语言的兄弟地区，也采用了不同的 DBCS 编码方案。

在1994年之前，也就是Unicode编码出现之前，大陆地区的中国人想使用台湾软件，还必须加装一套支持 BIG5 编码的“倚

天汉字系统"才可以用，装错了字符系统，显示就会乱码。

## 2、ISO（国际标准化组织）组织着手解决编码不统一的问题

舍弃了所有地区性编码方案，重新搞了一个包括了地球上所有文化、所有字母和符号的编码！叫做"Universal Multiple-Octet Coded Character Set"，（字面意思：全球通用的多八位字节字符集，中文名称为：**统一字符集**）简称 **UCS**，俗称 "Unicode"。

### 【备注】

octet 英[pk'tet] 美[a:k'tet]

n. 八位位组，八位字节；

Unicode是计算机科学领域里的一项业界标准,包括字符集、编码方案等。Unicode 是为了解决传统的字符编码方案的局限而产生的，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。1990年开始研发，**1994**年正式公布。Unicode编码的出现，其实就是为了适应全球化的发展，便于**不同语言之间的兼容交互**，而ASCII编码无法完成这个任务。

## （二）、Unicode编码的特点

### 1、Unicode 开始制订时，计算机的存储器容量极大地发展，空间再也不成为问题。

- Unicode字符集中，ISO 直接规定**必须用两个字节**，也就是**16位**来统一表示所有的字符；
- 对于ASCII码中的字符，Unicode 保持其原编码不变，但是将其长度由原来的**8位扩展为16位**，而其他文化和语言的字符则全部重新统一编码；
- 由于"半角"英文符号只需要用到低8位，所以其高8位永远是0来补齐，因此这种大气的方案在保存英文文本时会多浪费一倍的空间。

### 2、Unicode 是用两个字节来表示为一个字符，总共可以组合出65536不同的字符

这大概已经可以覆盖世界上所有文化的符号。但是Unicode对汉字支持其实依然不够好，因为简体和繁体汉字总共有六七万个汉字，而UCS-2最多能表示65536个，所以Unicode只能排除一些几乎不用的汉字，好在常用的简体汉字也不过七千多个。为了能表示所有的汉字，ISO已经准备了UCS-4方案，给Unicode增加了UCS-4规范，就是用四个字节来表示一个字符，这样就可以保存42亿个不同的字符了！

### 3、Unicode字符集中包含中文字符20902个。其中第一个是汉字“一”（位置编号为19968，十六进制表示为：**\u4E00**），最后一个汉字“𪚩”（位置编号40869，十六进制表示为：**\u9FA5**）。请大家牢记：**[\u4E00-\u9FA5]**。它表示Unicode字符集中第一个中文字符到最后一个中文字符的意思，换句话说它是所有中文字的表达式。这在以后的工作中经常用到。

### 【备注】

“𪚩”字同【吁】字。

- 一拼音xu1。
  - 叹息：长～短叹。
- 二拼音yu4【吁】的简体字
  - 为某种要求而呼喊：呼～。
- 三拼音yu1
  - 象声字，吆喝牲口停止前进的声音

### 4、Unicode编码与GBK系列编码最大的区别

- Unicode编码虽然与GBK都是双字节编码，但是Unicode编码中每个字符都是两个字节，即便是原ASCII码中的字符也是两个字节；而GBK编码中依然保留原ASCII码中的字符为单字节，除此之外的字符才是双字节。

5、Unicode 在制订时没有考虑与任何一种现有的编码方案保持兼容

因此 GBK 与Unicode 在汉字的内码编排上完全是不一样的，没有一种简单的算术方法可以把文本内容从Unicode编码和另一种编码进行转换，这种转换必须通过查表来进行。

五、Unicode字符集的字符编码规则

(一)、UTF编码

1、事实证明，对可以用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。

一个ASCII中的数字和字母，明明可以用八位0和1就表示清楚，Unicode非要用16位来表示。那么多余的那8位（也就是高字节）用0来补齐，对ASCII码来说高字节的0毫无用处，白白浪费了空间。

为了保证在网络传输中减少不必要的流量浪费，提升文本传输速度，出现了一些中间格式的字符集，他们被称为**统一的转换格式**，即UTF系列编码（Unicode Transformation Format）。**UTF是Unicode字符集的编码方式**。

常见的UTF格式有：**UTF-8**、UTF-16以及 UTF-32。无论是UTF-16还是UTF-8都是处理Unicode字符集的，但是它们的编码规则不相同。

(二)、UTF-16编码

1、UTF-16比较好理解，就是任何字符都用两个字节来保存。

- Unicode在定义时就是两个字节表示一个字符，因此常常将Unicode与UTF-16等同对待；
- UTF-16**编码效率最高**，字符到字节相互转换更简单，进行字符串操作也更好。它适合在**本地磁盘和内存之间**使用，可以进行字符和字节之间的快速切换，如Java的内存编码就采用UTF-16编码；
- UTF-16不够经济，如果文档中都是英文字母，那么这个做就非常浪费。明明用一个字节就能保存的偏用两个字节来保存。尤其在现在的网络带宽有限的今天，使用UTF-16编码会增大网络传输的流量，很没必要；
- UTF-16编码不适合在网络之间传输，因为网络传输容易损坏字节流，一旦字节流损坏将很难恢复。

(三)、UTF-8编码

1、utf-8编码是Unicode字符集最常用的编码方式。

- UTF-8比UTF-16的解码要复杂。UTF-8编码是**可变字节编码**；
- 文本读取是一个字节一个字节来的读取，然后再根据字节中开头的bit标志来识别，然后确定几个字节是一个字符单元；
- UTF-8字符集中，原Unicode前128个字符是**单字节编码**（实体编号在127以内），编号在128至2047的是**双字节编码**（2的11次方=2048），编号在2048之后就是**三字节编码**；
  - 一个字节代表一个字符：第一个字节为0。
  - 两个字节代表一个字符：第一个字节的前3位是110，第2个字节的前2位是10，说明两个字节代表一个字符；
  - 三个字节代表一个字符：第1个字节的前4位是1110，第2个字节的前2位是10，第3个字节的前2位是10，说明三个字节代表一个字符。如下图：

0x0080 - 0x07FF	110xxxx 10xxxxxx
0x0800 - 0xFFFF	1110xxxx 10xxxxxx 10xxxxxx

- UTF-8文件中，ASCII码占一个字节，中文字则占**三字节长度**；
- 相比较UTF-16而言，UTF-8更适合**网络传输**。
  - 对ASCII字符采用单字节存储，最大程度低节省文件传输时的流量大小；
  - 单个字符损坏也不会影响后面的其他字符，编码安全性强；
  - 在编码效率上介于GBK和UTF-16之间；
  - 因此，UTF-8在编码效率上和编码安全性上做了平衡，是理想的中文编码方式。

2、UTF-8文件的解码过程

(1)、建议使用的软件工具：

- 进制转化可以通过[在线进制转换](https://tool.lu/hexconvert/)工具实现。
- 查看十六进制代码建议使用：UltraEdit文本编辑器
- 网页制作建议使用：HBuilder前端开发工具

(2)、举例说明计算机对UTF-8文件的解码过程

有一个UTF-8编码的文本，文本内容如下：

“aá—”

分别是英文字母 “a”，法语字母 “á”，中文汉字 “—”。

#### 一、获取十六进制编码为：

61 C3 A9 E4 B8 80

#### 二、计算二进制编码：

01100001

11000011

10100001

11100100

10111000

10000000

此时，能看出该文本共有六个字节。到底哪几个字节是一个字符单元呢？

#### 三、根据UTF-8编码规则将字节分组后：

01100001

11000011

10100001

11100100

10111000

10000000

#### 四、重新计算，得出对应Unicode字符集的二进制编码为：

00000000 01100001

00000000 11100001

01001110 00000000

#### 五、计算机从Unicode字符集中反查出字符为：

a

á

—

【如何知道上述二进制编码对应的字符是什么呢？】

#### A、十进制编号：

97

225

19968

#### B、知道Unicode字符的10进制编号后，反查该字符呢？

在html文件中利用html实体编号就可以查出来。

- &#实体编号；

&#97;&#225;&#19968;

输出：aá—



#### (四)、UTF-8+编码 (UTF-8 with BOM)

##### 1、BOM (Byte Order Mark, 字节序列标记)

如果用记事本把一个文本文件另存为UTF-8编码方式的话, 用ultraEditor打开这个文件, 切换到十六进制编辑状态就可以看到开头的EF BB BF了。这是个标识UTF-8编码文件的好办法。如果接收者收到以EF BB BF开头的字节流, 就知道这是UTF-8编码的文件。所以UTF-8可以用BOM来表明编码方式。

其实EF BB BF的二进制编码为:

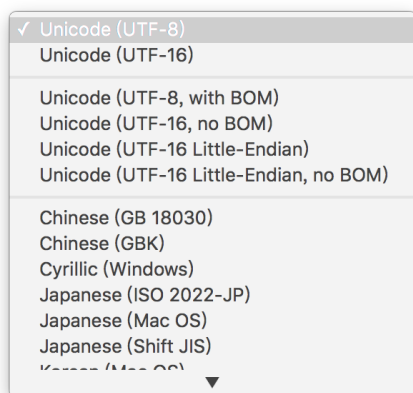
EF 11101111  
BB 10111011  
BF 10111111

转成Unicode二进制编码为: 1111110111111111

十进制编号为: 65279

在html页面上输出为: 空格

##### 2、UTF-8编码的文件中, BOM占三个字节。



#### (五)、UTF-16 with BOM编码

##### 1、保存的文件类型为UTF-16, 那么开头的字节流是: FF FE (表示UTF-16)。

FF FE的二进制编码为: 11111111 11111110

Unicode二进制编码同上。

十进制编号为: 65534

在html页面上输出为: 空格

##### 2、UTF-16编码的文件中, BOM占两个字节。

#### 六、动手练习: “奇怪的联通现象”

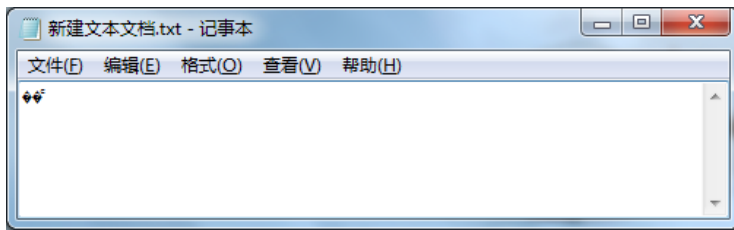
##### (一)、测试步骤:

1、请同学们动手试验一下。有个很著名的奇怪现象: 当你在 windows 的记事本里新建一个文件, 输入"联通"两个字之后, 保存, 关闭, 然后双击文件打开。观察到什么了吗?

文字消失, 取而代之的是几个乱码!

再试试输入“力挺联通”, 保存后打开会如何呢?

莫非联通得罪了微软?

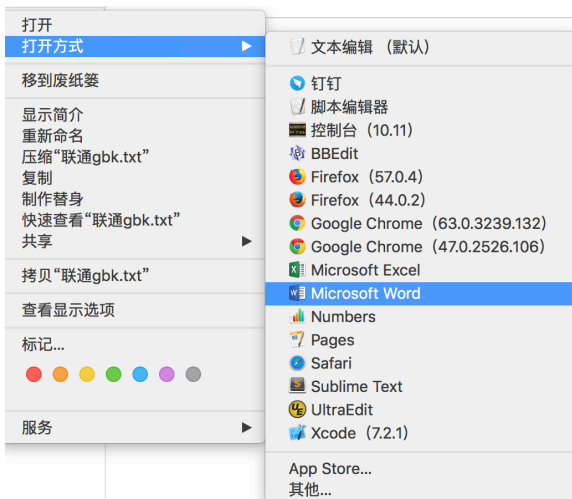


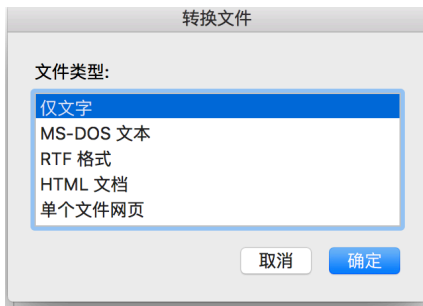
## (二)、解答

- 其实出现乱码的主要原因是：GB2312编码与UTF-8编码产生了编码冲撞，导致编码误解，从而触发了错误的文件打开方式所引起的；
- 当新建一个文本文件时，记事本的编码默认是ANSI，如果在ANSI的编码输入汉字，那么他实际就是GB2312系列的编码，在这种编码下，"联通"的十六进制编码是：

```
C1 1100 0001
AA 1010 1010
CD 1100 1101
A8 1010 1000
```

- 虽然保存的是GB2312编码，但是编码后的二进制数据正好和UTF-8的格式吻合，所以“记事本”把这些字符（"联通"）当做UTF-8的编码去解码，所以显示的结果会出现问题；
- 如果输入中文"爱联通"，保存文件后关闭。再次打开，则不会出现乱码问题，因为中文"爱"在编码表中对应的二进制数据不符合UTF-8的格式，所以记事本使用GB2312来解码，后面的"联通"当然也使用GB2312解码，所以就不会出现乱码。总之，只要GBK文本的二进制编码不与UTF-8吻合，就不会被误解为UTF-8文件，也就不会使用错误的打开方式来打开了；
- 非常奇妙的是输入“力挺联通”，还会出现乱码。原因是“力挺”这两个字的GB2312二进制编码也恰好与UTF-8吻合；
- 换一种文本编辑器来打开，或许问题就不会出现。如果没有专业的文本编辑器，甚至使用word或浏览器来打开，也能正常显示。





## 七、其它编码或字符集

### (一)、ANSI编码

ANSI是一种字符编码标准，其实就是各个国家或地区的国标。对于中国地区就是GB2312编码，中国台湾地区就是BIG5编码，对于日本就是JIS编码，对于美国自然就是ASCII码等等。

windows操作系统中的记事本，默认保存的编码就是ANSI编码（在中国就是GB2312）

### (二)、ISO-8859-1字符集

- 标准ASCII码只有128个字符，显然是不够用的，于是在ASCII码基础上又制定了一系列标准用来扩展ASCII编码，它们是ISO-8859-1~ISO-8859-15。其中ISO-8859-1涵盖了大多数西欧语言字符，应用的最广泛。
- ISO-8859-1仍然是单字节编码，它总共能表示256个字符。
- Linux操作系统默认的就是ISO-8859-1编码；而Win32操作系统的机器默认是GB2312编码。

### (三)、结束语

常见的字符集有ASCII、ISO-8859-1、GB2312、GBK、Unicode等，它们都可以被看作是不同的文化的大字典。字符编码它们规定了字符转化的规则，按照这个规则就可以让计算机能正确地识别字符。

目前存储中文的编码格式很多，例如GB2312、GBK、UTF-8、UTF-16这几种格式都可以存储汉字，那到底选择哪种编码格式来保存文件呢？这就要综合考虑、有所取舍，要考虑到到底是存储空间重要还是编码效率更重要，从而正确选择编码格式。