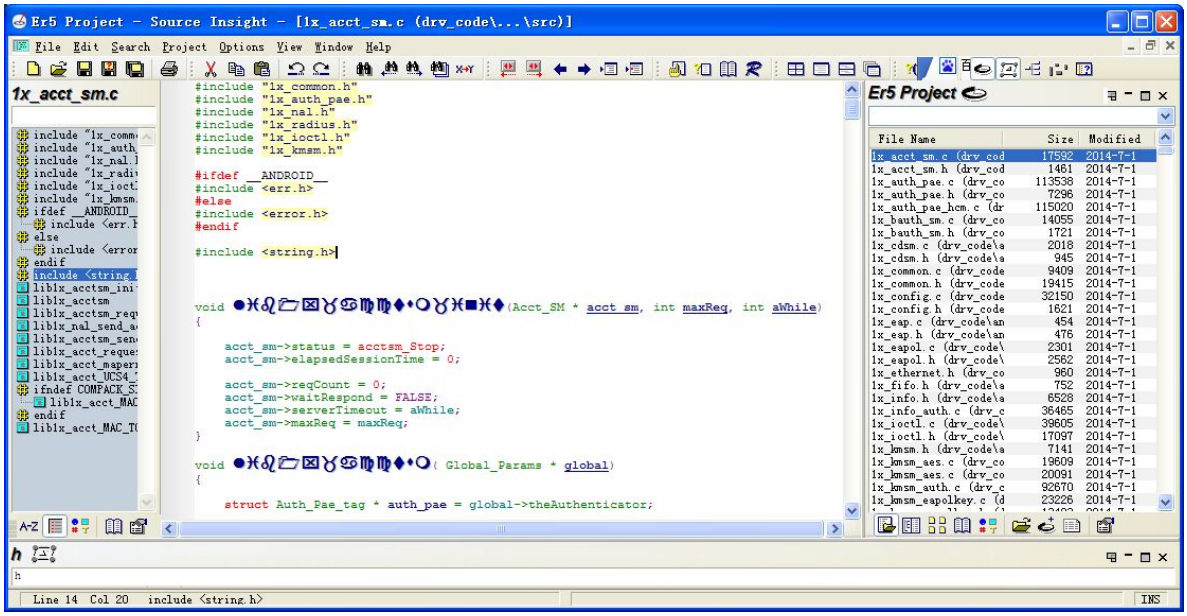
跟涛哥一起学嵌入式 26：深入浅出计算机编码、乱码问题

文档说明	作者	日期
来自微信公众号：宅学部落(armLinuxfun)	wit	2020.3.8
嵌入式视频教程淘宝店：https://wanglitao.taobao.com/		
联系微信：brotau(宅学部落)		

很多新手在编写程序、使用软件打开文档或者浏览网页时，经常遇到乱码显示、全角半角的问题。

```c
#include "led.h"

//////////////////////////////////////////////////////////////////////////////////
//ALIENTEK mini STM32
//LED
//@ALIENTEK
//www.openedv.com
//:2012/9/2
//V1.0
//
//Copyright(C) 2009-2019
//All rights reserved
//////////////////////////////////////////////////////////////////////////////////

//LED IO
void LED_Init(void)
{

 GPIO_InitTypeDef  GPIO_InitStructure;

 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOD, ENABLE);

 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;                 //LED0-->PA.8
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;          //
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;         //IO50MHz
 GPIO_Init(GPIOA, &GPIO_InitStructure);                    //
 GPIO_SetBits(GPIOA,GPIO_Pin_8);                           //PA.8

 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;                 //LED1-->PD.2
 GPIO_Init(GPIOD, &GPIO_InitStructure);                    //IO
 GPIO_SetBits(GPIOD,GPIO_Pin_2);                           //PD.2
}
```

网上也有很多解决的方法，大部分都是跟编码有关：比如Unicode、UTF-8、ASCII码、GB2312...，令人眼花缭乱，今天就给大家理一理它们之间的关系。



计算机只认识0和1这两个数字，我们输入的程序代码、文字都要经过编码，然后才能被计算机识别、解析和存储。早期的计算机环境是主要是英文，我们对构成英文的这些基本字母：拉丁字母编码就可以了，比如ASCII码。

ASCII码使用一个8位的单字节数据来编码电脑中常用的各种字符，如

- 拉丁字母：A、B、...、Z，a、b、...、z
- 数字：1、2、3、4、5、6、7、8、9
- 标点符号：逗号、句号、省略号
- 控制字符：回车符、换行符、空格符、制表符等

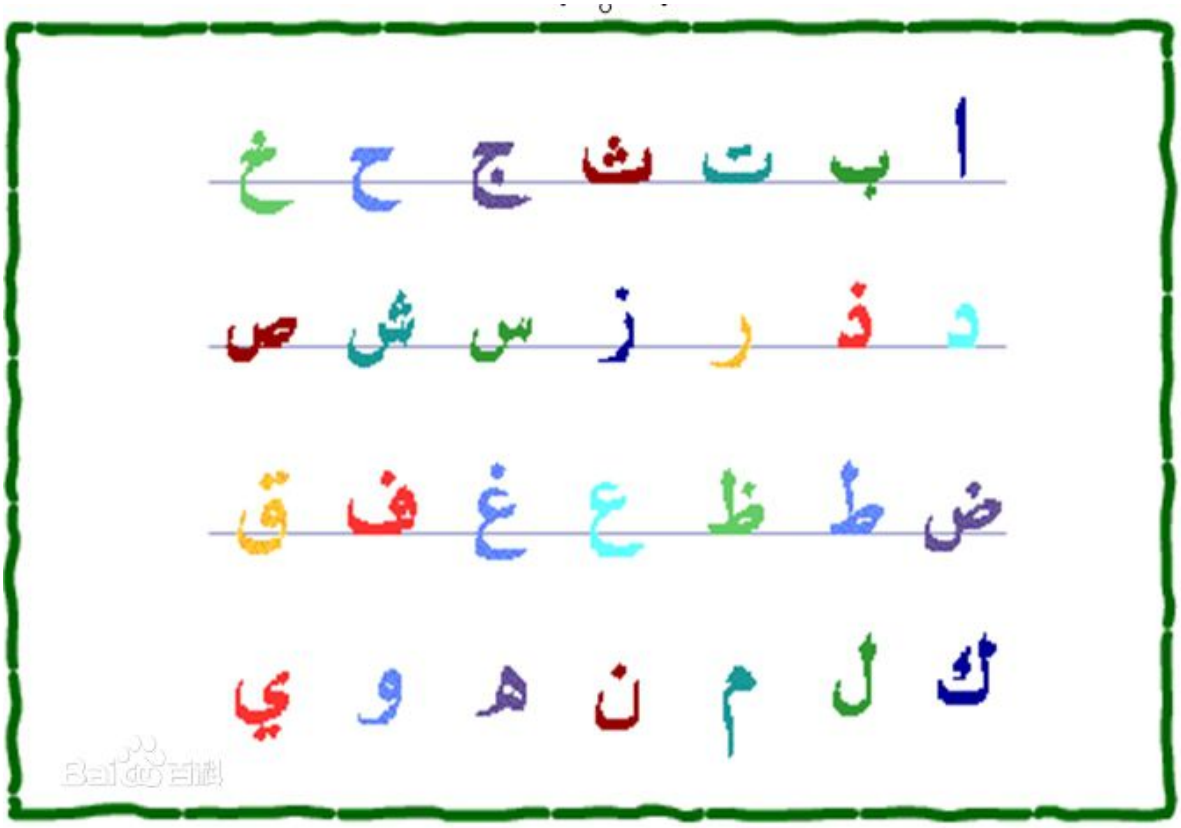
二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形
0010 0000	32	20	(空格)(sp)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x
0011 1001	57	39	9	0101 1001	89	59	Y	0111 1001	121	79	y
0011 1010	58	3A	:	0101 1010	90	5A	Z	0111 1010	122	7A	z
0011 1011	59	3B	;	0101 1011	91	5B	[	0111 1011	123	7B	{
0011 1100	60	3C	<	0101 1100	92	5C	\	0111 1100	124	7C	|
0011 1101	61	3D	=	0101 1101	93	5D	]	0111 1101	125	7D	}
0011 1110	62	3E	>	0101 1110	94	5E	^	0111 1110	126	7E	~
0011 1111	63	3F	?	0101 1111	95	5F	_				

ASCII码使用单字节的 bit0 ~ bit7 ，可以表示128个英文常用的拉丁字母和各种控制字符，这在英文环境下足够用了，随着计算机的普及，每个国家或地区都有自己个文字，这就给计算机的显示的麻烦，计算机中没有其他文字的编码，遇到这些文字，肯定没办法解析和显示了，显示的可能是一片乱码。

1. 世界三大字母

为了显示各国语言文字，我们需要对世界上各种语言做些分类。世界上的语言很多，主要可分为两类：象形型文字和字母型文字。象形型文字如汉字，除此之外，绝大部分语言文字都是字母型文字，基本上都是基于以下三大字母表去构建的。

- 拉丁字母：英语、法语、德语、意大利语、荷兰语、西班牙语、汉语拼音
- 阿拉伯字母：阿拉伯语、波斯语、维吾尔文
- 斯拉夫字母：俄语、乌克兰语、波兰语、白俄罗斯语、吉尔吉斯、乌兹别克、新蒙古语



А аз	Б бу́ки	В ве́ди	Г глаго́ль	Д добро́	Є есть	Ж живе́те
Ѕ зело́	З земля́	И и́же	І и	К ка́ко	Л лю́ди	М мысле́те
Н наш	О он	П поко́й	Р рцы	С сло́во	Т тве́рдо	Оу ук
Ф ферт	Х хер	Ѡ от, оме́га	Ц цы	Ч червь	Ш ша	Щ шта
Ъ ер	Ы еры́	Ь ерь	Ѣ ять	Ю ю	Ꙗ я	Ѧ юс малый
Ж юс большой		Ѯ кси	Ѱ пси	Ѳ фита́	Ѵ и́жица	

古希腊作为欧洲文明的起源，拉丁字母和斯拉夫字母都起源于希腊字母。希腊字母广泛用于数学、物理、生物、化学、天文等学科，如大家熟悉的α（Alpha）、β（Beta）、Ω（Omega）、Δ（delta）。后期经东正传教士传播到斯拉夫民族区并加以改造，就变成了斯拉夫字母。罗马人引进希腊字母后，稍加改变就成了拉丁字母。拉丁字母是世界上最流行，英语、法语、德语、西班牙语，甚至我们使用的汉语拼音都是使用拉丁字母，再加上早期的计算机主要在欧美，所以早期的计算机字符编码使用拉丁字母也就不奇怪了。

由于希腊字母在很多科研领域中的广泛应用，为了显示这些希腊字符，ASCII码进行了扩展了字符集，由原来的128个扩展到了256个：增加了希腊字母、特殊的拉丁符号以及一些表格符号、计算符号等。

ASCII编码简单点理解，其实就是一个字符集，每个字符通过编码，可以很方便地在计算机上被识别和存储。ASCII码的缺陷是使用单字节存储，最多也就知道编码256个字符，容量有限，尤其是各国都有自己的语言文字，比如中文，常用的就有近3000个汉字。再使用单字节编码存储肯定不行，需要扩充这些字符集。

2. GB2312编码

以微软操作系统为例，基本上世界各国都在使用它，都要显示自己的文字，比如我们要使用中文版的操作系统，要显示中文，怎么办？微软采用的方案是：各国采用各自的编码方案。以中文为例，我们有上万的汉字需要编码、存储，采用的是GB2312编码：0~127单字节编码表示原来的拉丁字母A~Z、a~z等，从127往后，每两个字节表示一个汉字。高低字节的编码方式可以编码6000多个常用汉字，除此之外，还把数学符号、罗马希腊字母、阿拉伯字母、俄文字母、日文的平假名、片假名都编进去了，就连ASCII表中原有的数字、字母、标点符号都使用双字节重新编码，这就是我们平常所说的全角字符，127号以下的那些单字节字符叫半角字符。GB2312编码可以看作是对ASCII的扩展。

3. GBK标准

中文除了简体，还有繁体字，也需要对这些繁体字进行编码。早期台湾地区使用BIG5编码对繁体字进行编码，也是采用双字节存储。随着电脑的普及，国内少数民族也要使用电脑，各个民族也有自己的语言系统。为此，GB2312字符集不断扩充，不断加入新的字符编码，于是就产生了GDB编码，并逐渐成为中文编码的标准。根据这个标准，可以将不同汉字进行编码构成字库，计算机想显示汉字，根据编码到字库去查就可以了。早期的计算机内存、存储资源有限，将字库固化到硬件ROM中，插到计算机上就可以了，这就是汉卡。《征途》老板史玉柱，当年就是靠这个汉卡起家的，赚得第一桶金，登上人生巅峰。现在的计算机一般不适用汉卡了，改用软件字库代替，直接存放到硬盘就可以了。

4. Unicode编码

各国都使用自己的编码方案，搞出一套自己的编码标准。用户在安装好Windows系统后，设置成本国语言就可以正常使用Windows了，可以正常显示本国的文字。在Windows系统中，简体操作系统使用的GBK，繁体操作系统使用的是BIG5，各个地区的本地编码方案作为不同语言版本的Windows的ANSI编码标准。但这种编码方案很容易出问题，随着互联网兴起，各国网民使用浏览器浏览网页时，浏览他国的网页时，如果本地字库没有编码这些网页的外语字符就很容易乱码。为了解决这个问题，ISO国际标准化组织废除了所有的地区性编码方案，重新搞了一套包括地球上所有语言、字母、字符的编码：Universal Multiple-Octet Coded Character Set，简称Unicode编码，又叫国际码。

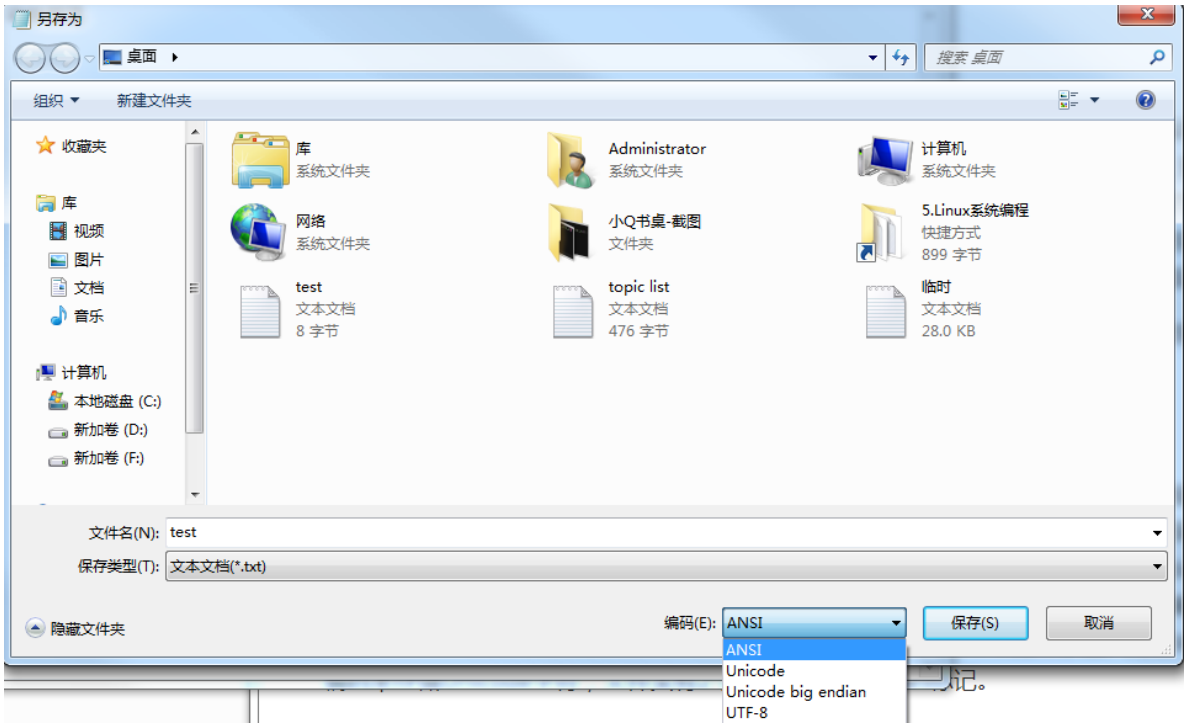
Unicode编码使用双字节来编码字符，一共可以编码65536个字符，这足以容纳地球上所有的语言文字和字符了，而且可以把所有的语言都编进去，全世界通用，多好！

5. UTF-8编码

Unicode编码作为国际码，解决了各国编码冲突问题，但是缺陷还是有的：浪费存储空间。比如原来的英文字符编码，单字节就可以了，现在是双字节，编码后的文件体积足足增大了一倍，不利于网络传输。为此，基于Unicode编码标准，UTF-8编码在存储上做了改进：采用变长字节(1~6个字节)来存储Unicode字符，原来的ASCII码采用单字节存储；希腊字母、斯拉夫字母采用2字节存储；汉字采用3字节存储。Linux环境下一般采用UTF-8编码存储文件，采用UTF-8编码存储的文件一般在文件头会有3个字节的UTF-8编码标记。而在Windows下，一般使用UTF-16编码来存储Unicode字符，文件头有2个字节的UTF-16编码标记。

6. 文件编码实验

我们在Windows下创建一个文本文件，输入4个汉字：宅学部落。保存文件分别保存为不同的编码格式：ANSI、Unicode、UTF-8，查看文件大小，分别为8字节、10字节、15字节。如果输入英文字符：wang，再分别保存并查看各个文件大小，大小分别为：4字节、10字节、7字节。



通过实验我们可以看到，使用UTF-8编码汉字，每个汉字3个字节，生成的文件体积比较大。因此很多中文操作系统下，经常还是有很多人使用GBK标准编码的。为了区分各种编码，一般在文件头会有几个字节说明该文件的编码方式，比如UTF-8文件编码存储的文件头部会有3个隐藏字节(0xEF 0xBB 0xBF)标记UTF-8编码，UTF-16文件头有2个字节(FE FF或FF FE)用来标记UTF-16编码方式，这种标记数据一般称为BOM头。

在Windows下使用记事本，如果采用Unicode存储，默认是自动给文件添加BOM头的。而在Linux下的文本文件虽然默认使用UTF-8标准，但是编码生成的文件一般是不带BOM头的，这也是很多新手在Windows下用记事本编写程序或者脚本，然后拷贝到Linux系统中运行，发现总是错误的原因。现在高级点的文本编辑器，如sublime、UltraEdit、notepad++等，都支持“UTF-8 无BOM”保存方式，编辑保存的文件更适合跨平台保存和运行。

7. 小结

以上给大家分享了不同语言文字、各种程序源文件、各种文本文档在计算机中如何编码和保存的小知识。不同的操作系统、不同的软件在存储字符到文本文件时，不仅编码方式不同，而且还会有BOM头的差异。理解了这些基本原理和细节后，大家在以后的编程中再遇到类似的问题，就迎刃而解了。

注嵌入式、Linux精品教程：https://wanglitao.taobao.com/
嵌入式技术教程博客：http://zhaixue.cc/
联系 QQ：3284757626
嵌入式技术交流QQ群：475504428
微信公众号：宅学部落(armlinuxfun)