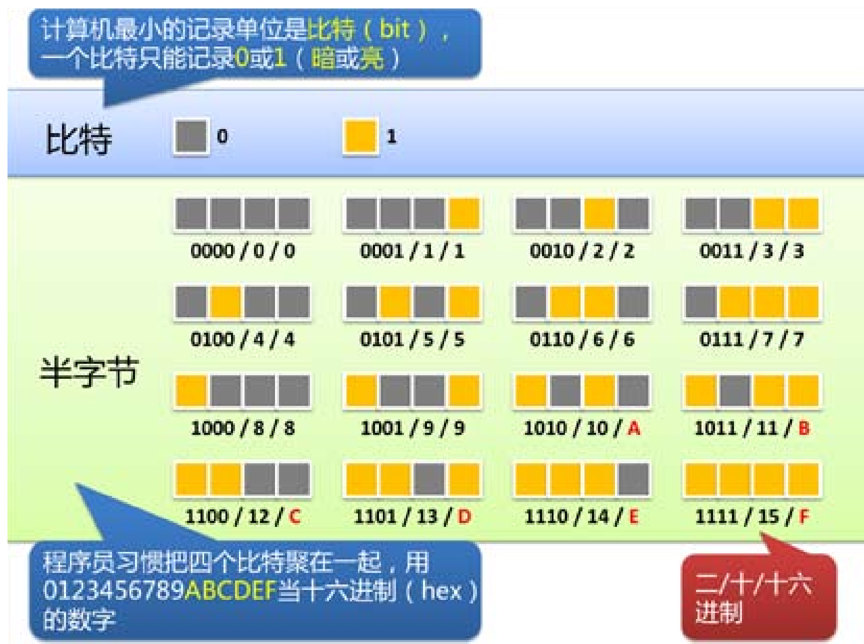
互动出版网 China-Pub.com
专业图书网上第一专营店
www.china-pub.com 独家提供样章

第4章

字符编码

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章



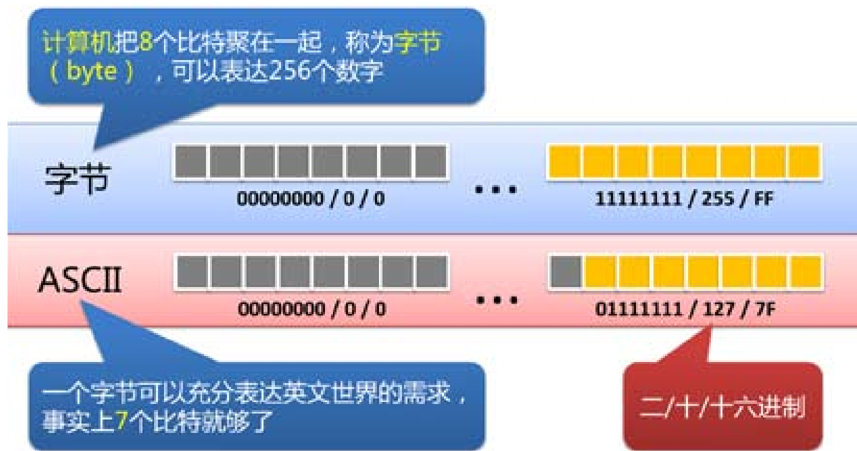
人类采用十进制，数字是 0–9。计算机是二进制的世界，只有 0 与 1 两种选择，称为比特（bit）。你可以想象一个灯泡就是一个比特，亮代表 1，不亮代表 0。

十进制的 0 写成二进制是 0，十进制的 1 写成二进制是 1，十进制的 2 写成二进制是 10，十进制的 3 写成二进制是 11，十进制的 4 写成二进制是 100，十进制的 256 写成二进制是 100000000。随着数值增大，二进制的长度增长得很快，十进制的 256 只需要 3 个数字，转成二进制却需要 9 个数字。二进制（比特）只适合计算机，不适合人类使用。

于是我们把四个比特结合在一起，姑且称为半字节，半字节能表示的数字范围提升到 16。而阿拉伯数字最高只能表达到 9，至于 10 ～ 15 分别用英文字符 ABCDEF（不区分大小写）表示。

写代码或者写文章时，我们很少使用二进制，一般使用十进制或者十六进制。为了区分写出来的数字是哪种进制，我们习惯在十六进制前面加上 0x，而十进制数字前不加任何东西。例如：0x20 相当于 32。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章



计算机把两个半字节结合在一起，称为字节（byte）。一个字节有 8 个比特，数字范围变成 0~255，也就是 0x00~0xFF。

英文字母大小写，加上 10 个阿拉伯数字，共 62 个，加上各种符号，就超出 64（2 的 6 次方）了。所以最早我们用 7 个比特（2 的 7 次方）来表示字符（character），因此制订了一个标准，叫做 ASCII。通常用一个字节来代表一个字符（不过这就浪费了一个比特）。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

绿色表示ASCII码 | 灰色表示不可显示字符 | 白色表示可显示字符 | ASCII就是英文世界的字符编码

00	空	10		20		30	0	40	@	50	P	60	`	70	p
01		11		21	!	31	1	41	A	51	Q	61	a	71	q
02		12		22	"	32	2	42	B	52	R	62	b	72	r
03		13		23	#	33	3	43	C	53	S	63	c	73	s
04		14		24	$	34	4	44	D	54	T	64	d	74	t
05		15		25	%	35	5	45	E	55	U	65	e	75	u
06		16		26	&	36	6	46	F	56	V	66	f	76	v
07	哔	17		27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18		28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19		29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A		2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C		2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D		2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E		1E		2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F		1F		2F	/	3F	?	4F	O	5F	_	6F	o	7F	删

这就是完整的 ASCII 字符编码表。绿色的部分是 ASCII 的编码（这里用十六进制数字表示），灰色与白色的部分是编码对应的字符，其中灰色的是不可显示字符，白色的是可显示字符。

只有可显示字符才能被打印出来。可显示字符的范围是在 0x20 与 0x7E 之间。注意，0x7F 是不可显示字符。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

^后紧接着一个可显示字符，形成转义串，会被转成一个不可显示字符（下一页有范例）

00	^@或^(null)	10	^P	70	p
01	^A	11	^Q	71	q
02	^B	12	^R	72	r
03	^C	13	^S	73	s
04	^D	14	^T	74	t
05	^E	15	^U	75	u
06	^F	16	^V	76	v
07	^G	17	^W	77	w
08	^H	18	^X	78	x
09	^I或^-或^(tab)	19	^Y	79	y
0A	^J或^/	1A	^Z	7A	z
0B	^K	1B	^[	7B	{
0C	^L	1C	^\	7C		
0D	^M	1D	^]	7D	}
0E	^N	1E	^!	7E	~
0F	^O	1F	^_	7F	^~

那些不可显示的字符，我们看不到，那么又要如何输入呢？我们可以通过转义串（escape sequence）的方式使用它们，就像是一种代号。编码为 0x00 的字符用 `^@` 表示，编码为 0x01 的字符用 `^A`（大小写皆可）表示等，如上表所示。

有些不可显示字符的转义串不止一种，例如编码为 0x09 的字符就有三种表示法，可以是 `^I`、`^-`、`^(TAB)`（大小写皆可）。

不可显示字符中，目前最常用的是 0x0A 与 0x09。输入 0x0A 有换行的效果，输入 0x09 会跳到下一个表格定位点（对交互环境来说，表格定位点就是 8 的倍数位置）。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
>> char-a: #"a"
== #"a"

>> print char-a
a

>> beep: #"^G"
== #"^G"
>> print beep
（发出一声哔）

>> to-hex/size to-integer beep 2
== #07

>> print "A^/B^/C"
A
B
C
```

这是一个可显示字符，打印（print）后看得到

字符的写法是这样的，前面是#"，后面是"

不可显示字符的转义串，打印后看不到，但可能会有其他效果。

打印^G字符，会发出一声“哔”

to-integer把字符转成整数

此字符的ASCII码

to-hex把整数转成十六进制

/size 2表示只保留最后两个数字

打印此字符串时，ABC之间有两次换行，是^/转义串的效果

字符串（字符串在一起）写法是这样的，前后都是"

字符的表达方式是在字符（或者字符的转义串）本身前后加上英文双引号，然后再在这个整体的前面加上 #。

打印可显示字符（例如 #"a"），很简单地在界面上就出现该字符。如果打印不可显示字符，又会如何？不同的不可显示字符有不同的效果，以 0x07 字符来说，效果就是发出一声“哔”。对 0x0A 字符来说，就是换行。

我们可以通过 to-integer 把字符转成整数，然后用 to-hex 把整数转成看起来像十六进制的值 #0000000000000007，但我们希望只取得最后两个数字，所以通过修饰字 /size 2 来改变 to-hex 的行为，使得 to-hex 只保留最后两个数字。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

写个程序用来分析文件内容。
文件名：DUMP.REB

```
01 REBOL[]
02 bin: read to-file system/script/args
03 forskip bin 16 [
04   ascii: copy " "
05   repeat i 16 [
06     either ch: bin/:i [
07       prin remove rejoin [ to-hex/size ch 2 " "]
08       append ascii either all [ ch >= 33 ch <= 126 ]
09         [ to-char ch ] [ #"." ]
10     ] [
11         prin "-- " append ascii #" "
12     ]
13   ]
14   print ascii
15 ]
16
17
```

若不在33到126之间，就不是可显示字符，用 . 表示。32是空格，在此也视为不可显示

本页代码目前看不懂不用担心

让我们写一个稍微长一点的程序。这个程序文件名为 dump.reb，它可以用来分析文件内容。把文件内容输出到界面上，界面分成左右两部分，左边是文件内容的十六进制数据，右边则把数据当成 ASCII 呈现出来。右边遇到不可显示的字符（包括空格 0x20）时就用一个点代替。

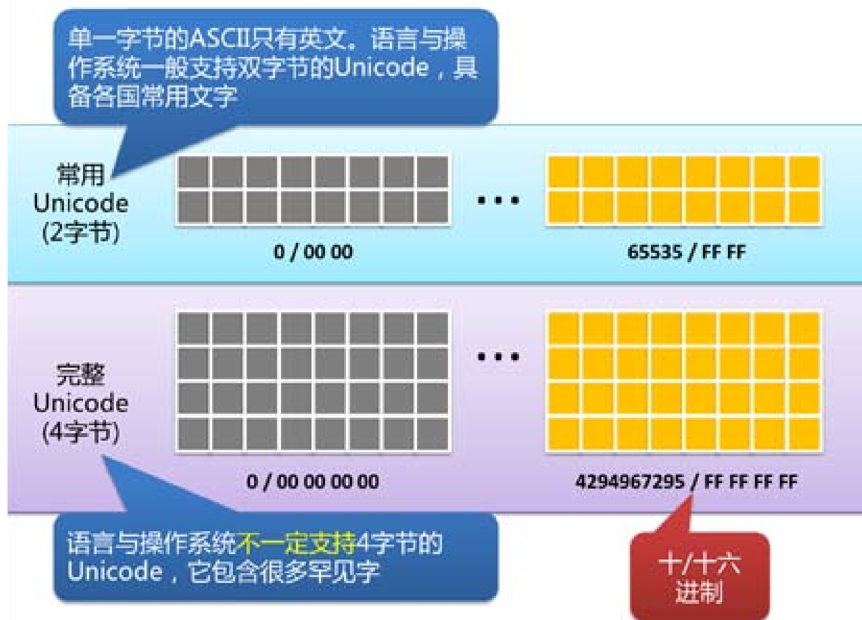
关于此代码的其他细节，在此不解释。

不如就用 dump.reb 来分析 dump.reb 自己吧？为了指定被分析的对象是 dump.reb，我们为 do 函数加上一个修饰字 /args。这个修饰字的出现，使我们可以通过参数的方式指定 dump.reb 是要被分析的文件。

此例子中，文件一开始居然不是 REBOL 文件头，而是多了三个可疑的字节：0xEF、0xBB、0xBF。这三个字节称为字节次序标记，用来表示此文件是采用 UTF-8 编码格式，这是这三个字节唯一的目的，我们可以不用理会它。如果没有这三个字节，也没关系。

仔细观察一下，换行的地方都会出现两个不可显示的字符，分别是 0x0D 与 0x0A。这是因为我是在 Windows 下用记事本写的代码，而 Windows 默认采用 0x0D 与 0x0A 这两个连续字符当一次换行。如果你是在 Linux 或者苹果的计算机上写代码，可能只需要 0x0A 就代表换行，不需要 0x0D。

互动出版网
China-Pub.com
专业图书网上第一专营店
www.china-pub.com
独家提供样章



世界上不是只有中文与英文，还有许多其他国家的语言，有一个名为 Unicode（统一码、万国码）的组织制订了一个标准，包含各种语言字符，每个 Unicode 字符都有一个编号，这个编号就称为 Unicode 码点。最常用的字符都在 Unicode 码点 0x0000~0xFFFF 中，所以需要两个字节。不管英文、中文或其他语言，一律使用两个字节表示。

另外还有一个更完整版本的 Unicode，用到 4 个字节的空间，可以容纳四十多亿个不同的字符，连一些康熙辞典上没有的字符都具备。4 字节的编码虽然比较完整，但用处不大，且为了这些使用概率极低的字符要让文本体积大幅度膨胀，大多数的时候不值得。为此，大多数的语言与操作系统当今支持的是两个字节的 Unicode。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

REBOL只支持双字节Unicode，不支持四字节Unicode

Unicode码点与UTF-16BE是一样的

		维	生	素	C
Unicode码点		7E F4	75 1F	7D 20	00 43
编码	UTF-16BE	7E F4	75 1F	7D 20	00 43
	UTF-16LE	F4 7E	1F 75	20 7D	43 00
	UTF-8	E7 BB B4	E7 94 9F	E7 B4 A0	43

REBOL 只支持UTF-8，不支持UTF-16与其他编码

UTF-8编码是变动长度的，英文需要一个字节，中文需要三个字节

大多数在 ASCII 之后出现的字符编码（encoding），都会以兼容于 ASCII 为目标，毕竟 ASCII 在英文世界是主流。想在兼容于 ASCII 的情况下，能够使用 Unicode 所有的文字？这就要采用变动长度的编码 UTF-8 了。

UTF-8 编码遇到 ASCII 中的字符，会采用一个字节编码；遇到某些字符（包括欧洲一些国家语言的字符）采用两个字节编码；遇到某些文字（尤其中日韩文字）采用三个字节编码，甚至某些字符采用四个字节编码。

对于正常的REBOL代码来说，使用标准ASCII字符的机会比中文字符高。所以采用UTF-8编码，可以让代码长度缩短，又兼容于 ASCII。因此 REBOL 规定脚本文件一律是 UTF-8 编码。

你可以到 Unihan 数据库网站（http://www.unicode.org/charts/unihan.html）搜索某个字符的 Unicode 信息，这里会列出该字符的 Unicode 码点、UTF-16LE 编码、UTF-16BE 编码、UTF-8 编码等信息。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

我们可以在代码中用转义串的方式产生 Unicode 字符。以 `#"^(4F20)"` 为例来说，这就代表 Unicode 码点为 0x4F20 的字符，也就是 `#"传"` 这个字符。这种表示法必须在圆括号内使用十六进制数字的 Unicode 码点。

在 REBOL 中，每个值都属于某种数据类型（datatype）。字符的数据类型就是 `char!`。通过 `type?` 函数就可以得知某个值的数据类型（第 9 章会详细说明数据类型）。

我们可以把多个字符组在一起，成为字符串（string），字符串的表达方式与字符类似，但不需要前面的 `#`，且不限定字符长度是 1。

字符串内也可以使用转义串，以上图中的 `"维生素C"` 字符串为例，可以写成 `"^(7EF4)^(75EF)^(7D20)C"` 或 `"^(7EF4)^(75EF)^(7D20)^(43)"`。

我们可以通过 `to-binary` 函数，将字符串转成 UTF-8 格式的二进制数据，这种转换叫做文字编码。也可以通过 `to-string` 的方式把 UTF-8 格式的二进制数据转成字符串，这种转换叫做文字解码。

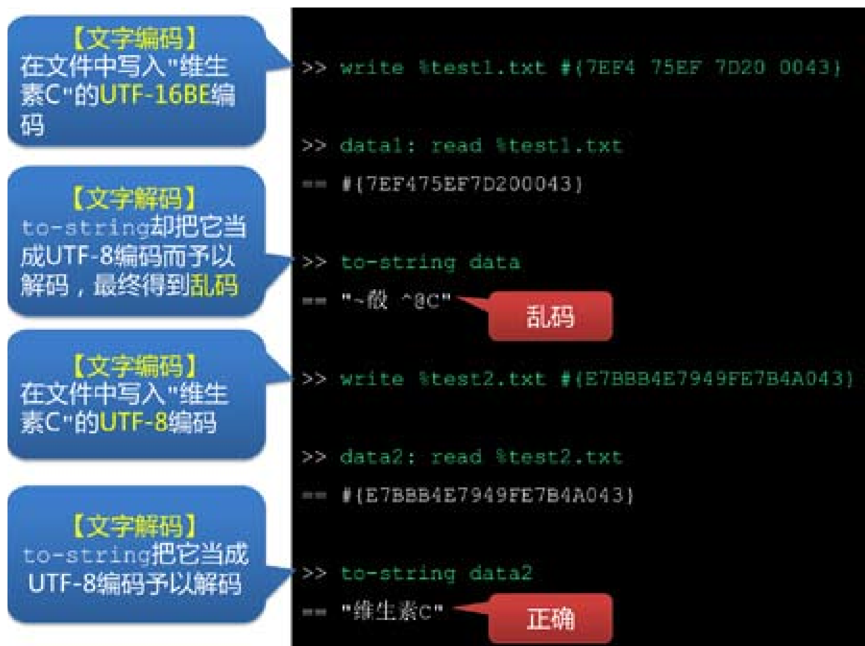
在执行 REBOL 代码时，内存中保存这些字符串的数据，最好每个字符采用统一的长度，这样会比较好处理。像 UTF-8 这种变动长度的编码方式，就不适合。内存里最好是 UTF-16BE 或者 UTF-16LE，或者 ASCII 编码（如果全都是英文）这类每个字符相同长度的编码。

假设 REBOL 解释器在内存中采用的就是 UTF-16BE 编码，我们说这是内部格式。

在保存或传输数据时，为了节省空间与加快速度，UTF-8 会是比较好的编码选择。因此 REBOL 对于文件一律采用 UTF-8 编码，我们说这是外部格式。

将内部格式转成外部格式，称为编码（encoding）。将外部格式转成内部格式，称为解码（decoding）。我们可以采用 `to-binary` 函数把文字编码成 UTF-8，采用 `to-string` 把 UTF-8 解码成内部的 UTF-16BE。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章



编码与解码必须要一致，或者至少要兼容，否则就会出现乱码。

在这个例子中，在文件 test1.txt 中写入 " 维生素 C" 的 UTF-16BE 编码。通过 read 函数读出文件中的原始数据，发现一切正常，正是我们刚刚写入的数据。然后通过 to-string 转换这些原始数据，就出问题了：乱码！这是因为 to-string 会把原始数据当做 UTF-8 而予以解码。

再做一次实验，这次写入的是 " 维生素 C" 的 UTF-8 编码。通过 read 函数读出文件中的原始数据，最后通过 to-string 解码，一切正常。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

最后的叮咛：REBOL 解释器只会用 UTF-8 纯文本方式为脚本文件解码，然后才执行。保存脚本文件时，一定要用兼容于 UTF-8 的方式编码（有无文件头的 BOM 皆可）。如果你的文件脚本是纯英文，那么以 ASCII 保存也行，因为 ASCII 兼容于 UTF-8。

☑ 我了解脚本文件的编码格式UTF-8

如果你确定做到了，到本篇开始“学习目标”处打钩。