中文信息处理

汉字编码

引自 李正华

苏州大学

2015 年 9 月 24 日

为什么要编码？

为什么要编码？

- 将文字存储在计算机中（数字化）

为什么要编码？

- 将文字存储在计算机中（数字化）
- ASCII

为什么要编码？

- 将文字存储在计算机中（数字化）
- ASCII
- ASCII: American Standard Code for Information Interchange (美国信息交换标准码)

为什么要编码？

- 将文字存储在计算机中（数字化）
- ASCII
- ASCII: American Standard Code for Information Interchange (美国信息交换标准码)
- ISO: 国际化标准组织

为什么要编码？

- 将文字存储在计算机中（数字化）
- ASCII
- ASCII: American Standard Code for Information Interchange (美国信息交换标准码)
- ISO: 国际化标准组织
- ISO IEC 646:1991 编码标准（ISO 7-bit coded character set for information interchange）

ASCII（ISO/IEC 646:1991）图形（可见）符号

- 7位二进制数，定义128个字符：
  - 94个图形字符（可显示字符）
    - ‘0’–’9’:　30H-39H
    - ‘A’-’Z’:　41H-5AH
    - ‘a’-’z’ :　61H-7AH

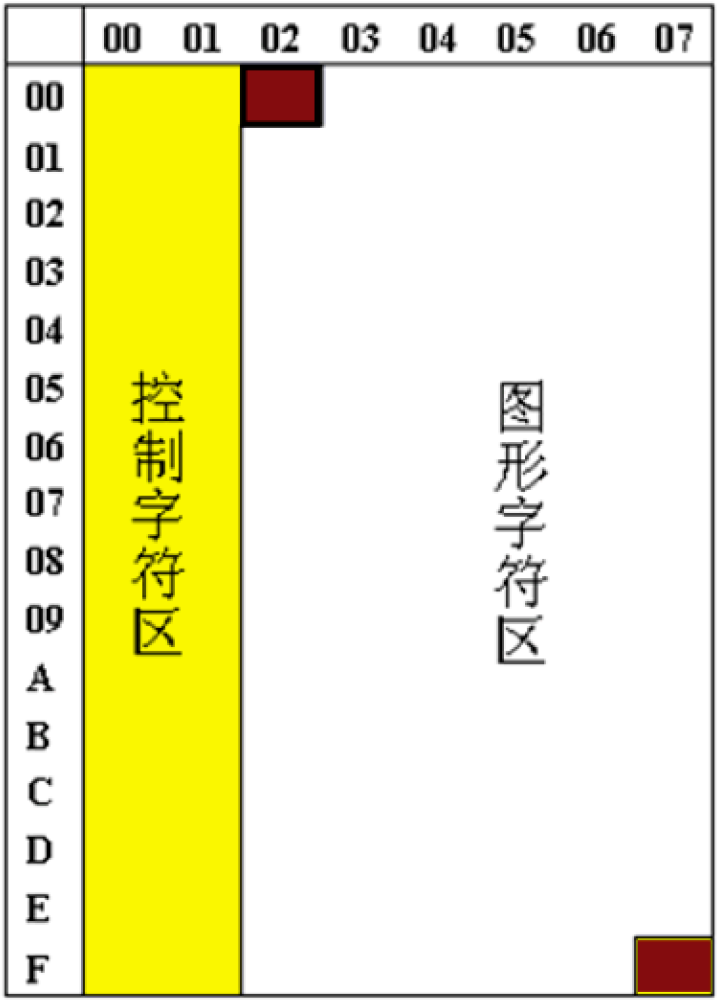
ASCII 码		字符	ASCII 码		字符	ASCII 码		字符	ASCII 码		字符
十进制	十六进制		十进制	十六进制		十进制	十六进制		十进制	十六进制	
032	20		056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(	064	40	@	088	58	X	112	70	p
041	29	)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[	115	73	s
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D	]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	J	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	|
053	35	5	077	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	⌂

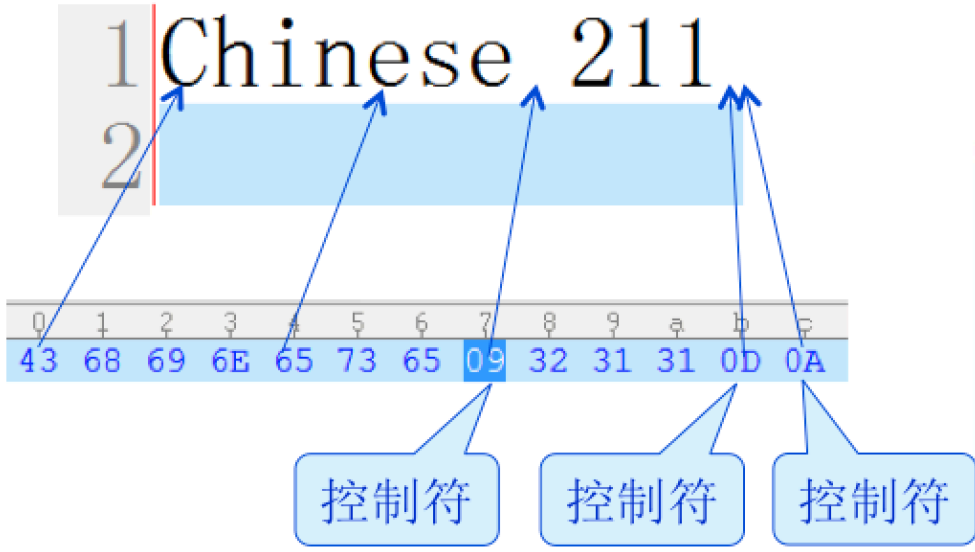
ASCII（ISO/IEC 646:1991）控制及其它符号

- 7位二进制数，定义128个字符：
  - 94个图形字符（可显示字符）
    - ‘0’–’9’:　30H-39H
    - ‘A’-’Z’:　41H-5AH
    - ‘a’-’z’ :　61H-7AH
  - 30个控制字符
    - 00-19H
  - 1个空格字符
    - 20H
  - 1个Del（删除）符
    - 7FH

ASCII码		字符	控制字符	意义	ASCII码		字符	控制字符	意义
十进制	十六进制				十进制	十六进制			
000	00		NULL		016	10	►	DLE	
001	01	☺	SOH		017	11	◄	DC1	
002	02	☻	STX		018	12	↕	DC2	
003	03	♥	ETX		019	13	‼	DC3	
004	04	♦	EOT		020	14	¶	DC4	
005	05	♣	ENQ		021	15	§	NAK	
006	06	♠	ACK		022	16	▬	SYN	
007	07	•	BELL	振铃	023	17	↨	ETB	
008	08	◘	BS	退格键	024	18	↑	CAN	
009	09		HT	定位键	025	19	↓	EM	
010	0A		LF	line feed	026	1A	→	SUB	档案结束
011	0B	♂	VT	home	027	1B	←	ESC	escape
012	0C	♀	FF	form feed	028	1C	L	FS	向右键
013	0D		CR	carriage return	029	1D	↔	GS	向左键
014	0E	♫	SO		030	1E	▲	RS	向上键
015	0F	☼	SI		031	1F	▼	US	向下键

ASCII（ISO/IEC 646:1991）控制及其它符号：示例





ISO 8859：扩展 ASCII 码

ISO 8859：扩展 ASCII 码

- 7 位扩展为 8 位，增加 128 个码元

ISO 8859：扩展 ASCII 码

- 7 位扩展为 8 位，增加 128 个码元
- 增加一些欧洲国家的字母，主要为拉丁语系

为什么要对汉字编码？

为什么要对汉字编码？

- 将汉字存储在计算机中（数字化）

为什么要对汉字编码？

- 将汉字存储在计算机中（数字化）
- 汉字一共多少个？

为什么要对汉字编码？

- 将汉字存储在计算机中（数字化）
- 汉字一共多少个？
- 需要多少个码元？

几种汉字编码标准

- GB 编码（GB2312；GBK；GB18030）

几种汉字编码标准

- GB 编码（GB2312；GBK；GB18030）
- BIG5（繁体）

几种汉字编码标准

- GB 编码（GB2312；GBK；GB18030）
- BIG5（繁体）
- Unicode（ISO/IEC 10646）

Linux（或 Mac）下的几个命令

- iconv：文本编码转换

Linux（或 Mac）下的几个命令

- iconv：文本编码转换
  - iconv -l：查看当前 iconv 支持哪些编码

Linux（或 Mac）下的几个命令

- iconv：文本编码转换
  - iconv -l：查看当前 iconv 支持哪些编码
  - iconv -f ‘utf8’ -t ‘gbk’ txt-utf8 > txt-gbk

Linux（或 Mac）下的几个命令

- iconv：文本编码转换
  - iconv -l：查看当前 iconv 支持哪些编码
  - iconv -f ‘utf8’ -t ‘gbk’ txt-utf8 > txt-gbk
- hexdump txt-utf8（以 16 进制输出文件内容）

汉字 GB2312 编码中的区位码、交换码、机内码

- 区位码：汉字在 94×94 二维表中的位置，行（区）号，列（位）号

汉字 GB2312 编码中的区位码、交换码、机内码

- 区位码：汉字在 94×94 二维表中的位置，行（区）号，列（位）号
- 交换码：为了避开 ASCII 码中的控制码（00H-1FH），区位码的区号和位号都加上 020H

汉字 GB2312 编码中的区位码、交换码、机内码

- 区位码：汉字在 94×94 二维表中的位置，行（区）号，列（位）号
- 交换码：为了避开 ASCII 码中的控制码（00H-1FH），区位码的区号和位号都加上 020H
- 机内码：汉英混合文本中，为了避免和单字节的 ASCII 码混淆，交换码的两个字节最高位都改为 1

汉字 GB2312 编码中的区位码、交换码、机内码

- 1981 年国家颁布了 GB2312 汉字标准共有 6763 个, 其中一级 3755, 二级 3008, 还有 682 非汉字字符。并为每个字符规定了标准编码, 便于在计算机内部相互转换。作为 GB2312 标准只是定义了一张 94×94 的二维表。其中行为区号, 列为位号。这样可以利用区号和位号来找到其中的汉字。这种编码就是我们所说的区位码。

UTF16 编码

- 至少由两个字节表示一个字
- 0x0000-0xFFFF，两个字节
- 0x00010000-0x0010FFFF，四个字节

字节序：big/little endian

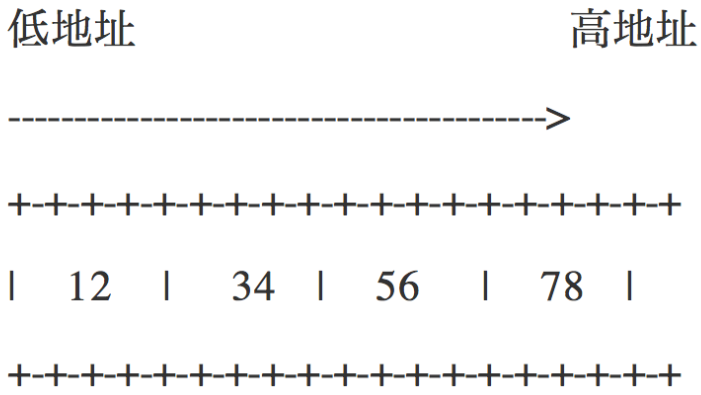
- 参考：
  http://blog.csdn.net/sunshine1314/article/details/2309655和
- http://baike.baidu.com/link?url=
  g9zAcbX9IFV9DrjTCgceyhgCkhszJGC1HUMEWLQ8bqlqmBw5yIO
  sJsj7RF4vdIIGZT90k-MnPWZI_

字节序：big/little endian

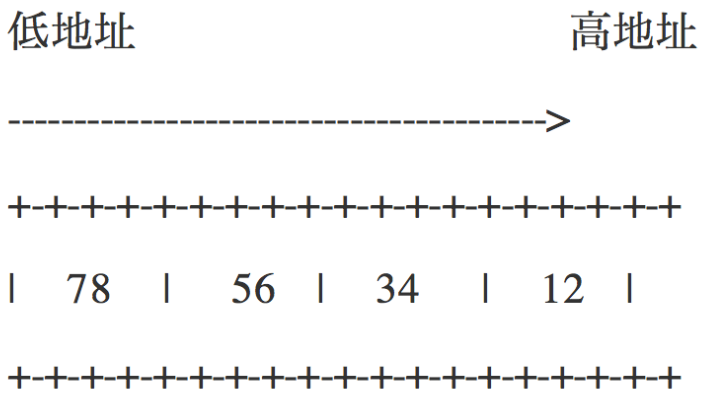
- 如何将大于一个字节的数据类型（如 int/double）存到内存中（寄存器）?
- big endian：低地址存高位，高地址存低位（hexdump 人看起来比较自然）
- little endian：相反
- 两种字节序有不同的 CPU 厂商支持，Java 统一采用 big endian，而 C/C++ 则和编译器所在的 CPU 相关
- 计算机程序处理时，两种字节序各有优势
- 网络协议全部采用 big endian，所以 big endian 又称为网络字节序
- 写通讯程序时（如机器内部进程间的通讯，如 C++ 和 Java 程序之间），要注意字节序的问题

图示

Big Endian



Little Endian



字节序：big/little endian

- 我的 Mac 下，用 hexdump a.txt 时（文件中含有一些 ascii 文本），似乎是 big endian
- 虚拟机 Ubuntu 下，似乎是 little endian
- hexdump 这个命令，其字节序取决于机器（操作系统?）设置
- hexdump -Cv a.txt（按 Byte 顺序输出）

字节顺序标记（BOM） Byte Order Mark

- 文本文件的具体存储方式
- 文本文件头插入几个字节（可选），表示文件的编码方式，及 BOM
- UTF-16 big endian: FE FF
- UTF-16 little endian: FF FE
- UTF-8: EF BB BF
- Mac 下，用 Sublime 选择 Save with encoding，可以有多个选项
- 随课件会给一些例子，大家可以用 hexdump 看。

2 个编程作业（共 6 分）

- 给一段汉语文本（可能含英文字母等 ASCII 符号），将文本切分为单个 character，并以空格隔开输出，最后输出一共有多少个 character（C 语言实现）
- 情况 1：GB 编码
- 情况 2：UTF8 编码