Character encodings

Low-level view: Byte streams

1 Byte = 8 bit, enough to encode $2^8 = 256$ characters

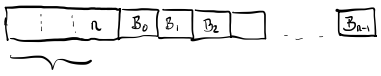
Standards:
- ASCII, ~ 1967, 7-bit
- various 8-bit extensions, most important ISO 8859
  16 versions 1987–2001, e.g. Latin-1 "Western European"

Issues:
- Barely works for single latin-like alphabets, difficult to mix languages
- 8 bit not enough for many languages (Chinese, arabic, ...)

Data model:

(1) Byte array ("C-style strings")
- A string is an array of bytes, direct indexing
- Convention: Termination with 0-character (0x00)
- Simple things are simple and fast, low memory overhead
- Everything else hard to get 100% right
- Memory safety hard to get right → security implementation
- Still used in system-level programming, small embedded systems

(2) Length-prefixed ("Pascal-style strings") (Python: bytes, bytearray)



- Makes a lot of string algorithms easier, faster, and more robust
- No special handling of 0x00
- Prefix-size is a compromise between maximal string length and memory requirements for small strings (rarely an issue nowadays)
- Strings are different from standard arrays

High-level view: Character streams "character" = Unicode code point

Standard:
- Unicode 1.0, 1991 16-bit
  Targets "modern" languages,
  $2^{14} = 16384$ "should be enough for all modern usage" (Becker '88)
  rest are "private code points"
- Unicode 2.0, 1996 beyond 16 bit
  17 planes, first plane (BMP) is, more or less, old unicode
  Codepoints U+0000 ... U+10FFFF ($\approx$ 1 million)
- Now at version 15.0, keeps changing/adding characters

Encodings:
- UTF-32: 32-bit per character
  often internal format, direct indexing possible!
- UTF-16: BMP via 16 bits, other planes as "surrogate characters"
- UTF-8: ASCII as 8-bit, BMP 1-3 byte surrogates, other planes 4 bytes
  empirically most efficient (even for Chinese web pages!)
  ASCII is valid UTF 8 !

Usage:
- Unix operating systems, WWW: UTF-8
- Windows: UTF-16
- In high-level programming languages (Python, ...), encoding details are hidden from programmer

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0 ^@ NUL NULL	1 ^A SOH START OF HEADING	2 ^B STX START OF TEXT	3 ^C ETX END OF TEXT	4 ^D EOT END OF TRANSM.	5 ^E ENQ ENQUIRY	6 ^F ACK ACKNOWL-EDGE	7 ^G BEL BELL	8 ^H BS BACKSP.	9 ^I HT CHARACT. TAB'TION	10 ^J LF LINE FEED	11 ^K VT LINE TAB'TION	12 ^L FF FORM FEED	13 ^M CR CARRIAGE RETURN	14 ^N SO SHIFT OUT	15 ^O SI SHIFT IN
1	16 ^P DLE DATALINK ESCAPE	17 ^Q DC1 DEVICE CONTROL1	18 ^R DC2 DEVICE CONTROL2	19 ^S DC3 DEVICE CONTROL3	20 ^T DC4 DEVICE CONTROL4	21 ^U NAK NEG.ACK-NOWLEDGE	22 ^V SYN SYNCHR. IDLE	23 ^W ETB END OF TRANS.	24 ^X CAN CANCEL	25 ^Y EM END OF MEDIUM	26 ^Z SUB SUBS-TITUTE	27 ^[ESC ESCAPE	28 ^\ FS INFO. SEP. 4	29 ^] GS INFO. SEP. 3	30 ^^ RS INFO. SEP. 2	31 ^_ US INFO. SEP. 1
2	 SPACE	! excl ! EXCLAM. MARK	" quot " QUOT. MARK	# num # NUMBER SIGN	$ dollar $ DOLLAR SIGN	% percnt % PERCENT SIGN	& amp & AMPER-SAND	' apos ' APOS-TROPHE	(lpar (LEFT PAREN.	) rpar) RIGHT PAREN.	* ast * ASTERISK	+ plus + PLUS SIGN	, comma , COMMA	- - HYPHEN-MINUS	. period . FULL STOP	/ sol / SOLIDUS
3	0 0 DIGIT ZERO	1 1 DIGIT ONE	2 2 DIGIT TWO	3 3 DIGIT THREE	4 4 DIGIT FOUR	5 5 DIGIT FIVE	6 6 DIGIT SIX	7 7 DIGIT SEVEN	8 8 DIGIT EIGHT	9 9 DIGIT NINE	: colon : COLON	; semi ; SEMI-COLON	< lt < LS.-THAN SIGN	= equals = EQUALS SIGN	> gt > GR.-THAN SIGN	? quest ? QUEST-ION MARK
4	@ commat @ COMM'IAL AT	A A	B B	C C	D D	E E	F F	G G	H H	I I	J J	K K	L L	M M	N N	O O
5	P P	Q Q	R R	S S	T T	U U	V V	W W	X X	Y Y	Z Z	[lsqb [LEFT SQ. BRACKET	\ bsol \ REVERSE SOLIDUS	] rsqb] RT. SQR. BRACKET	^ hat ^ CIRCUM'X ACCENT	_ lowbar _ LOW LINE
6	` grave ` GRAVE ACCENT	a a	b b	c c	d d	e e	f f	g g	h h	i i	j j	k k	l l	m m	n n	o o
7	p p	q q	r r	s s	t t	u u	v v	w w	x x	y y	z z	{ { L. CURLY BRACKET	| | VERTICAL LINE	} } R. CURLY BRACKET	~ ~ TILDE	127 ^? DEL DELETE

ASCII code table including entity references, control codes and Unicode names (1.1) Tom Gibara July 2014