ICS3U: Introduction to Computer Science

ASCII and Unicode

J. Garvin

ASCII

The *American Standard Code for Information Interchange* is a character-encoding scheme based on the Latin alphabet.

It specifies a numeric code for each character.

All computers that are ASCII-compatible will interpret these characters the same way.

ASCII

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

ASCII

The ASCII chart shown lists all of the printable characters. There are additional characters (whitespace, movement, computer codes) that are represented by codes 0-31.

Note that there are different codes for "A" (decimal 65) and "a" (decimal 97).

The computer has no idea what an "A" or an "a" is, or that they represent the same letter.

This is important to remember – a computer only understands electronic signals, and has no knowledge of our alphabet.

ASCII

Encoding strings of characters is done by replacing the individual characters with their corresponding codes (binary, in the computer's case).

For example, the word "Hello" has ASCII codes 72, 101, 108, 108 and 111.

In binary, this would be encoded as 01001000 01100101 01101100 01101100 01101111.

Parity and Data Transmission

Recall that all ASCII characters have decimal numbers between 0 and 127.

This means that all ASCII characters can be represented using 7 bits ($1111111_2 = 127_{10}$).

Since most modern computers use 8-bit bytes, this allows for the remaining bit to serve as a basic form of error-checking.

Parity refers to the *evenness* or *oddness* of a value.

Parity can be either *even* (the total number of 1s is even), or *odd*.

Parity and Data Transmission

Example: The ASCII character "A" is to be transmitted using odd parity.

"A" has a decimal value of 65, or 1000001 in binary.

The number of 1s in the 8-bit byte that is transmitted must be odd. Since there are two 1's in the binary representation of "A", the eighth bit becomes 1 to make a total of three 1's.

Therefore, "A" is transmitted as 11000001.

Parity and Data Transmission

Example: The ASCII character "=" is to be transmitted using even parity.

"=" has a decimal value of 61, or 0111101 in binary.

The number of 1s in the 8-bit byte that is transmitted must be even. Since there are five 1's in the binary representation of "=", the eighth bit becomes 1 to make a total of six 1's.

Therefore, "=" is transmitted as 10111101.

Unicode

In 1987, work began on another character-encoding system that would allow for a greater number of symbols and characters.

In 1991, members from Xerox, Sun, Apple, Microsoft, NeXT, and others developed the *Unicode Consortium*.

Unicode is a character-encoding system that uses up to 4 bytes (32 bits) to represent each character.

This means that Unicode can handle up to 4 294 967 296 characters. This is more than enough to handle most of the characters in all of the world's major languages.

In the official specifications, however, the number of characters is restricted to just over 1 million entries.

Unicode

Unicode

There are two main variants of Unicode that are supported by modern operating systems.

UTF-8 is the most prominent, and is the default character-encoding system on Linux/BSD/UNIX-style systems.

UTF-8 uses 1 byte (8 bits) to represent the standard ASCII characters, and up to 4 bytes (32 bits) to represent other characters.

UTF-16 is the default format in Windows. It uses 1 or 2 16-bit binary strings.

UTF-32 is also used, though rarely. It uses a single 32-bit binary string for each character.

Advantages and Disadvantages of Unicode

One advantage of Unicode is the availability of more characters, so multiple character-encodings are not needed if a document contains multiple languages.

Having one unified standard means that there is only one system to maintain.

Using up to 4 bytes per characters however, can result in large file-sizes and a greater amount of data that need to be transmitted.

Also, having a huge amount of characters can make finding the code for a particular character much harder.