이승용 - bongbongco@gmail.com

문자열이란 무엇인가.

1. 문자열이라 하면 떠오르는 것

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    char *string = "문자열이란 무엇인가";
    printf("%s\n",string);
    return 0;
}
```

```java
public class StringTest {
    public static void main(String[] args) {
        String string = "문자열이란 무엇인가";
        System.out.println(string);
    }
}
```

문자열이라 한다면 보통 자신에게 친숙한 언어의 형태를 빌려 'char*' 이나 'String' 과 같은 자료형을 떠올리게 됩니다. 다음 그림은 사람과 프로그래머, 컴퓨터 각자의 입장에서 문자열을 인식하고 처리하는 것에 대한 간략한 설명입니다.

※폰 노이만의 설계에 따라 메모리 밖에서는 어떤 자료도 읽거나, 실행할 수 없으므로 컴퓨터가 사용하기 위해서는 메모리에 저장되어야 합니다.메모리에는 비트가 존재하고, 비트의 집합인 바이트를 문자열로 인식하게 됩니다.

2. |0|1|1|0|0|0|0|1| : 97 혹은 'a'

0과 1로 저장된 메모리를 효율적으로 읽는 방법을 고민한 결과 자료형이라는 방법을 고안하게 됩니다. 저장된 자료를 어떻게 읽을 것인가를 결정하는 것이 바로 자료형입니다. 예를 들어 같은 메모리를 읽더라도 C 언어를 기준으로 int 나 char 중 어떤 자료형을 사용했느냐에 따라 어떻게 읽힐지가 결정됩니다.

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    char *string = "문자열이란 무엇인가";
    char testChar = 'a';
    int testInt = 97;

    printf("%s\n",string);
    printf("%c : %d\n",testChar, testChar);
    printf("%d : %c\n",testInt, testInt);
    return 0;
}
```

```
문자열이란 무엇인가
a : 97
97 : a
Program ended with exit code: 0
```

3. 문자집합

모든 문자를 메모리에 저장할 수 없으므로 문자집합을 만들게 된다. 컴퓨터의 설계는 유럽이 주도 했으나, 실제로 구현하고 발전된 쪽은 미국 이었습니다. 그 결과, 미국의 문자집합을 도입한 컴퓨터가 시장을 주도하게 됩니다.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

(※ ASCII TABLE 은 미국의 문자만 존재합니다.)
그리고 그 과정에서 “정해진 틀”이 바로 **1 Byte** 입니다.

3.1. 한글의 고통

한자 문화권의 국가들은 ASCII Table 로 자국의 문자 표현이 불가 하였습니다. 컴퓨터의 활용이 커짐에 따라 문자를 표현해야 하는 일이 발생하는 데, 이때 정해진 틀인 **1 Byte** 안에 문자를 정의하게 위해서 고민하게 됩니다.

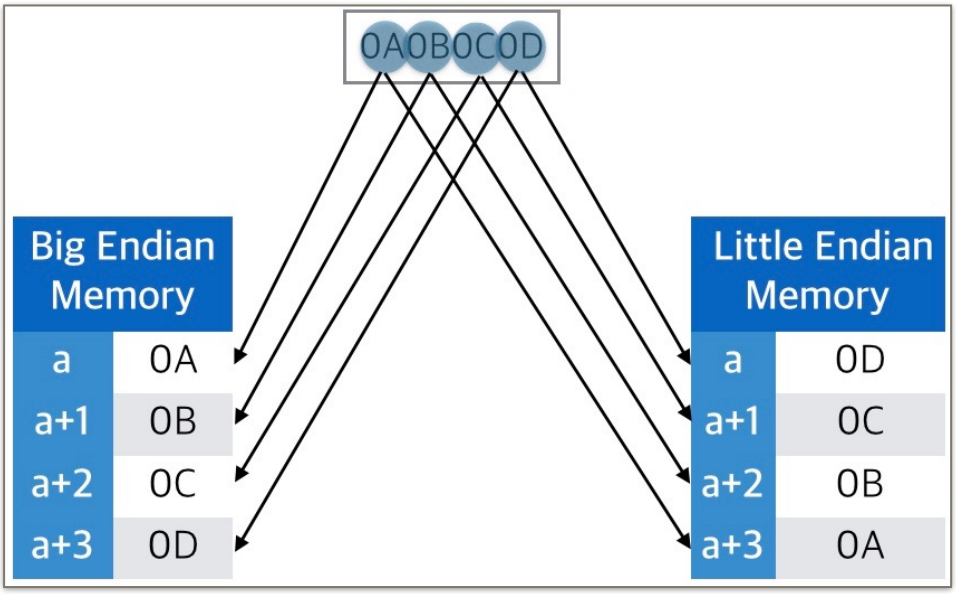
조합 형과 완성 형

한글 인코딩 방식을 두고 조합 형과 완성 형 중 어떤 것이 더 나은 방식인가 를 두고 논쟁을 벌이던 때가 있었습니다. 현재는 유니코드가 완성 형과 조합 형 모두 호환 가능하여 인하여 무의미 해졌습니다. 조합 형은 초성, 중성, 종성을 조합하여 모든 한글 문자를 표현할 수 있지만, 처리에 부담이 있고 다른 문자와 호환이 힘듭니다. 완성 형은 한글 한 음절에 코드를 부여하는 방식으로 한글의 원리를 반영하지 않는 다는 점과 일부 한글을 표현할 수없다는 단점이 있습니다. 이때 유행한 문장이 '찜차를 타고 온 펲시맨과 쑛다리 똠방각하' 입니다.

3.2. MBCS (멀티바이트 문자 집합)

1Byte의 한계에 부딪혀 정해진 틀을 2바이트로 늘리게 됩니다. 하지만 이러한 결정은 메모리에 담긴 바이트를 어떤 순서대로 인식하고 기록할 것인가? 라는 또 다른 고민을 만들게 됩니다. 모두 동일한 순서대로 읽는다면 상관없겠지만, 인텔 x86의 등장으로 앞에서 부터 읽을 것인가 뒤에서 부터 읽을 것인가 를 고려해야 되는 상황이 찾아오게 됩니다. 이렇게 읽고 쓰는 순서를 엔디언이라 합니다.

엔디언에 따른 바이트를 쓰거나 읽는 순서



그림에서 볼 수 있듯이 앞에서 부터(상위 바이트 부터) 읽고 쓰는 방식을 ‘Big Endian’, 뒤에서 부터(하위 바이트 부터) 읽고 쓰는 방식을 ‘Little Endian’ 이라고 부릅니다. 실제로 사용되는 대부분의 PC 가 Little Endian 방식을 사용하고 있으나, 네트워크 전송 시 Big Endian을 사용하기 때문에 Big Endian 방식과 Little Endian 방식을 혼동하지 않도록 주의합니다.

4. 인코딩

이 때 세상의 모든 문자를 수용하는 UNICODE 등장했습니다. 모든 문제가 해결된 듯 하였으나, UNICODE를 완벽하게 사용하기 위해서는 4Byte가 필요했습니다. 그래서 이를 최대한 줄인 UTF-8을 사용하게 됩니다.

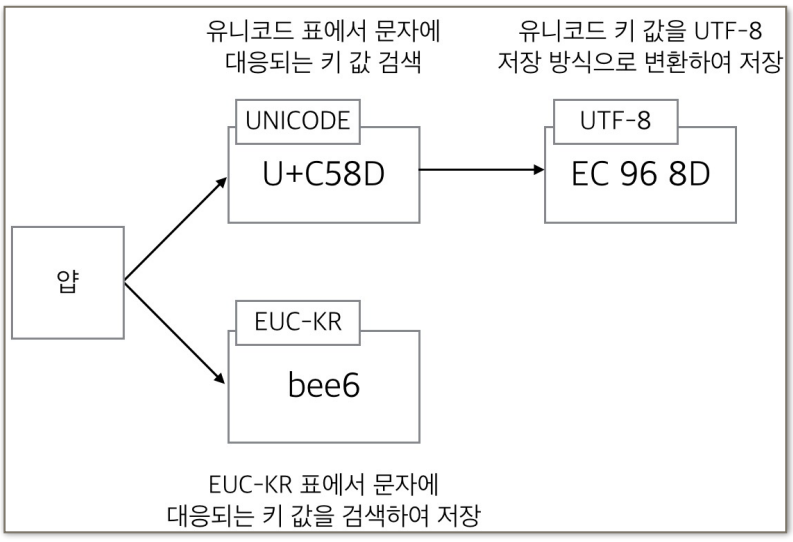
	영어	한글
EUC-KR	1byte	2byte
UCS-2	2byte	2byte
UTF-8	1byte	3byte
UTF-16	2byte	2byte
UTF-32	4byte	4byte

UNICODE

UNICODE와 인코딩 방식과 혼동 해서는 안됩니다. ASCII Table 에서 'a'와 97이 서로 대응관계에 있는 것 처럼 UNICODE 는 ASCII Table에서 포함하지 않는 전 세계의 문자를 특정한 키와 대응시켜 놓은 Table 인 것입니다. UTF-8은 인코딩 방식으로 UNICODE를 어떻게 표현하는 가를 결정합니다. 예를 들어 가변 바이트를 사용하는 UTF-8은 1 Byte로 표현이 가능한 'a'의 경우 0x61 과 같이 표현 할 것입니다. 그에 반해 UTF-16 은 2Byte(16 Bit)로 표현하므로 0x0061, UTF-32는 0x00000061으로 표현됩니다.

※ Little Endian을 사용하는 지, Big Endian을 사용하는 지에 따라 0x6100이 될 수도 있고 0x0061이 될 수도 있습니다.

EUC-KR vs UTF-8



한글 인코딩 방식은 크게 두 가지로 볼 수 있습니다. EUC-KR(혹은 EUC-KR이 확장된 CP949)과 UTF-8 입니다. 한글의 원리 상 조합 형 문자 인코딩 방식을 사용하는 것이 맞지만, 가장 많이 사용되는 OS인 윈도우에서는 완성 형을 기본 정책으로 채택하고 있어 문제가 발생하게 됩니다. CP949는 조합 형에서 만들 수 있는 모든 한글 문자를 포함하고 있지만 웹 서비스와 같은 경우 서로의 인코딩 방식을 동일하게 하여야 정상적으로 문자 표현이 가능합니다. UTF-8은 대표적인 조합 형의 유니코드 인코딩으로 가변 바이트 라는 점과 ASCII를 호환한다는 특징이 있습니다. EUC-KR 은 'KSC5601(2350자가 담긴 확장 한글 문자 집합)' 과 'KSC5636(ASCII를 토대로 하지만 역 슬러시를 원화로 바꾼 문자집합)' 이 결합된 형태로 구성되어있습니다. 또한 언어팩이 설치되어 있지 않아도 문자를 표현할 수 있는 유니코드와는 달리 한글을 사용하는 곳에서만 한글 문자를 표현할 수 있는 단점이 존재하여 인터넷 정보 전달 표준으로 사용하기에는 부적합니다.

5. 결론

문자열을 사용한다는 것은 다음 4 가지 과정을 내포하고 있습니다.

1. 메모리에 적재되는 것을 전제함
2. 어떤 문자 집합을 사용할 것인지 결정
3. 어떤 순서로 기록할 것인지 결정
4. 문자 집합의 키 값을 어떻게 인코딩하고 디코딩할 것인지 결정

참고 URL

EUC-KR 코드 표 : http://www.mkexdev.net/Community/Content.aspx?parentCategoryID=4&categoryID=14&ID=125

UNICODE 코드 표 : https://ko.wikipedia.org/wiki/유니코드_C000~CFFF

한글 인코딩의 이해 : http://d2.naver.com/helloworld/19187