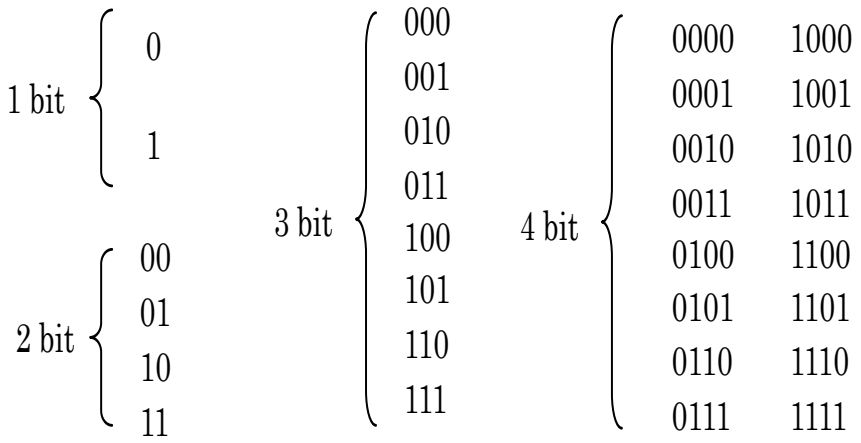
第24章　文字のデジタル化

- ✓ 情報量 bit
- ✓ ASCIIコード
- ✓ Byte
- ✓ テキストファイル

1. 情報量の単位　bit

デジタル化によって 0 と 1 とに表現された情報は、2進数のように 0 と 1 がいくつも連なった 01 列になっている。この 0 と 1 の個数のことを **bit**（ビット　bとも書く）という。2進数の 1010_2 は 4bit、11001110_2 は 8bit である。

一つの 01 列、つまり 1bit の情報について考えてみよう。1bit の情報とは、0 か1かしかない。次に二つの 01 列、つまり 2bit の情報について考えると、それは 00、 01、10、 11 という全部で4通りの情報がありうる。また、3 bit の 2 進数では 000、 001、010、011、100、101、110、111 の8通りの表現が可能だ。



bit 数が増えると表現可能なパターンが増える

このように bit 数が大きいほど、0 と 1 だけで表現できる情報が多くなる。

一般に n bit の 01 があれば、2^n 個の情報が表現できる

例えば四つの状態を０と１だけを使って表現するなら、2bit あれば可能だ。四つの記号 ♥ ♣ ♠ ♦ を 0 と1だけを使って表現するには、以下のような対応表を作ればよい。

♥	♣	♠	♦
00	01	10	11

これはトランプマークのデジタル化といってもよい。

八つの状態を 0 と 1 だけを使って表現するなら、3bit あれば可能だ。例えば、明日の天気を表すのに以下のような対応をさせてみる。

快晴	晴れ	曇り	雨	雷雨	雪	大雪	台風
000	001	010	011	100	101	110	111

このような対応表を作っておくと、3bit だけで明日の天気を表現できるようになる。これは天気のデジタル化といってもよい。

アルファベットは26文字あるので、26状態を表現するには、5bit あればよいことになる。そこで、例えば、以下のようにルールを決めることもできる。

A	B	C	・・・	Z
00000	00001	00010	・・・	11001

こういうデジタル化は、世界中が同じルールを使用しないと意味がない。

表 1 bit 数と情報の数

bit 数	表現可能な情報の数
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256

2. 文字のデジタル化

ここでは、アルファベットだけでなく、英語圏で使用されるすべての文字をデジタル化するためのルールについて学ぶ。

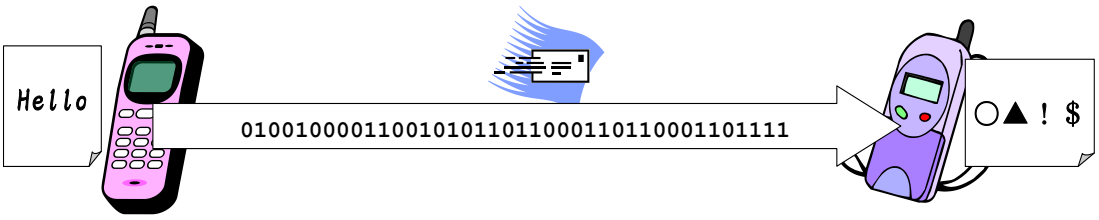
まず、電子メールで「Hello」という5文字を送ることを考えよう。この言葉をどうやってインターネットや電子メールは遠く離れた相手に伝えているのだろうか。まず「Hello」の 5 文字をコンピューターで扱える形に直すことが必要だ。コンピューターは0と1でしか情報を記録したり、通信したりすることができない。だから、文字を0と1だけを使って表現すること、すなわち文字のデジタル化が必要だ。

実は、Hello の 5 文字を世界共通のルールでデジタル化すると、以下の 40bit の情報になる。

0100100001100101011011000110110001101111

これならコンピューターで扱えるし、通信もできる。

この「Hello」という 5 文字を勝手なルールで 01 列に置き換えるわけにはいかない。受信した相手がこんな一見無意味に思える 40bit の01列を正しい元の 5 文字に戻すためには、お互いに共通の変換ルールが必要になる。



変換規則が同じでないと相手に伝わらない

共通の変換規則があれば、受信した側はこの 01 からなる 40 文字を解読して「Hello」と戻すことができる。実は、世界共通で以下のような1文字 8bit の対応関係が決められている。

H	e	l	o
01001000	01100101	01101100	01101111

3. ASCII コード

アルファベット文字を表すのに、8bit 必要なわけを考えてみよう。英数字とよく使う記号だけを考えて、文字の総数を数えてみると、以下の計算のように92種類もある。

アルファベット大文字・・・	２６種類
アルファベット小文字・・・	２６種類
数字・・・・・・・・・・・	１０種類
よく使う記号・・・・・・・	３０種類程度
合計	９２種類程度

それでは、92 種類の状態を表すのにいったい何 bit 必要であろうか。2 の n 乗を計算してみる。

$$2^4 = 16 \qquad 2^5 = 32 \qquad 2^6 = 64 \qquad 2^7 = 128 \qquad 2^8 = 256$$

2 の 6 乗では、64 状態が表現できるが、92 には足りない。2 の 7 乗なら 128 状態表現できるため、すべての英数字を表すのに十分である。したがって、7 bit あれば表現できることになる。

7 bit というのは、奇数であり扱いにくい。そのため、もう 1 bit 加えて 8 bit で１文字を表すことにする。無駄な 1 bit は使わなければよい。

8 bit = 英数字１文字

そこで、以下の表のように世界中の人々が共通で使えるように 8 bit の 01 パターンを文字に対応づけするように決めることにした。この対応づけは全世界共通であるため、この 01 パターンでインターネットなどの通信で世界中のコミュニケーションができるようになる。この対応関係を **ASCII（アスキー）コード**[1]という。

ASCII コードだけでなく、文字のデジタル化に使われる文字コード表はこの表のように 01 列で表記すると表が大きくなるため、実際には**16進数**を使ってもっと簡易的に表現されている。

[1] ASCII･･･American Standard Code for Information Interchange　規格の正式名称は ISO-8859-1 である。

表2　8ビットの01パターンと英数字の対応表

ASCII コード表

(スペース)	00100000	0	00110000	@	01000000	P	01010000	‘	01100000	p	01110000
!	00100001	1	00110001	A	01000001	Q	01010001	a	01100001	q	01110001
“	00100010	2	00110010	B	01000010	R	01010010	b	01100010	r	01110010
#	00100011	3	00110011	C	01000011	S	01010011	c	01100011	s	01110011
$	00100100	4	00110100	D	01000100	T	01010100	d	01100100	t	01110100
%	00100101	5	00110101	E	01000101	U	01010101	e	01100101	u	01110101
&	00100110	6	00110110	F	01000110	V	01010110	f	01100110	v	01110110
‘	00100111	7	00110111	G	01000111	W	01010111	g	01100111	w	01110111
(	00101000	8	00111000	H	01001000	X	01011000	h	01101000	x	01111000
)	00101001	9	00111001	I	01001001	Y	01011001	i	01101001	y	01111001
*	00101010	:	00111010	J	01001010	Z	01011010	j	01101010	z	01111010
+	00101011	;	00111011	K	01001011	[	01011011	k	01101011	{	01111011
,	00101100	<	00111100	L	01001100	¥	01011100	l	01101100	|	01111100
-	00101101	=	00111101	M	01001101	]	01011101	m	01101101	}	01111101
.	00101110	>	00111110	N	01001110	^	01011110	n	01101110	~	01111110
/	00101111	?	00111111	O	01001111	_	01011111	o	01101111		01111111

4. 情報量の単位　Byte

ASCII コード表のように、8 bit あれば、任意の英数字を表現でき、便利であることから、8 bit をまとめて一つの単位とし、1 **Byte**（バイト B とも書く）とする。今後、1 Byte は、英数字 1 文字分と考えてよい。

$$1\ \text{Byte} = 8\ \text{bit}$$

すなわち、

$$1\ \text{Byte} = \text{英数字}\ 1\ \text{文字}$$

と考えられる。

5. テキストファイルとサイズ

メモ帳を使って書いた何の飾りもない文章をそのまま保存すると、それら入力した文字をコード表により変換したのとまったく同じものがファイルとして保存される。このようなファイルを一般にテキストファイルと呼んでいる。

メモ帳を使って「Hello World」と書いてファイル名 hello.txt で保存する。このファイルは ASCII コードで変換された 01 列がそのまま保存される。そのファイルをメモ帳で開くと、その 01 列を ASCII コードだと認識して文字となってメモ帳に表示される。その関係を表3に示した。

表3 テキストファイル

メモ帳で編集	保存されたファイル　hello.txt
メモ帳では字の大きさや色などの書式情報は設定できない。	ハードディスクには「Hello World」という 11 の文字を単純に ASCII コードに変換した結果だけが保存される。
	「01001000011001010110110001101100011011110010000001010111011011110111001001101100011001 00」という情報がファイルとして保存される。このファイルをメモ帳で開くと再び「Hello World」となって表示される。

実際にメモ帳を使って文章を書き、保存されたテキストファイルのサイズを調べてみよう。

[ファイルサイズを調べる]

メモ帳にアルファベット数文字（例えば abcd）だけを書いて保存する。

そのファイルサイズを調べる（☞**演習4**）。

[バイナリーエディターでファイルの中身を見る]

メモ帳は、ファイル中の 01 列を ASCII コードによって変換された01列だと考え英数字に変えて表示してしまうため、実際にどのように格納されているかはわかりにくい。ASCII コード表によって変換をせずにファイルの中身をそのまま見ることができる便利なツール（バイナリーエディター）があるので、それを使用してファイルを閲覧してみよう（☞**演習5**）。

演習

1　日本語で使用される漢字やひらがなは7000文字程度である。もしこれを0と1だけで表現するとなると、何 bit 必要になるか？

2　自分の名前のローマ字表記をデジタル化してみよう。

スペースも 8bit で表現されることに注意。

例：　Toyo Eiko　⇒

010101000110111101111001011011110010000001000101

011010010110101101101111

自分の名前は何ビットで表現されますか？

自分の名前は何バイトで表現されますか？

3　簡単な英文メッセージを作成して、情報交換してみよう。

あまり長いと、変換が大変です。

手順1.　簡単な英文を考える　例：Hello World

手順2.　ASCII 表を使って01列に変換して紙に書く

010010000110010101101100011011000110111100100000010101110110111110111001001101100011001 00

手順3.　01列だけを、隣前後同士で交換する

手順4.　もらった01メッセージを解読する

もらったメッセージは、何 bit ですか？何 byte ですか？何文字ですか？

4　ファイルサイズの調べ方

保存してあるファイルサイズを調べる方法は

1．エクスプローラーを開く

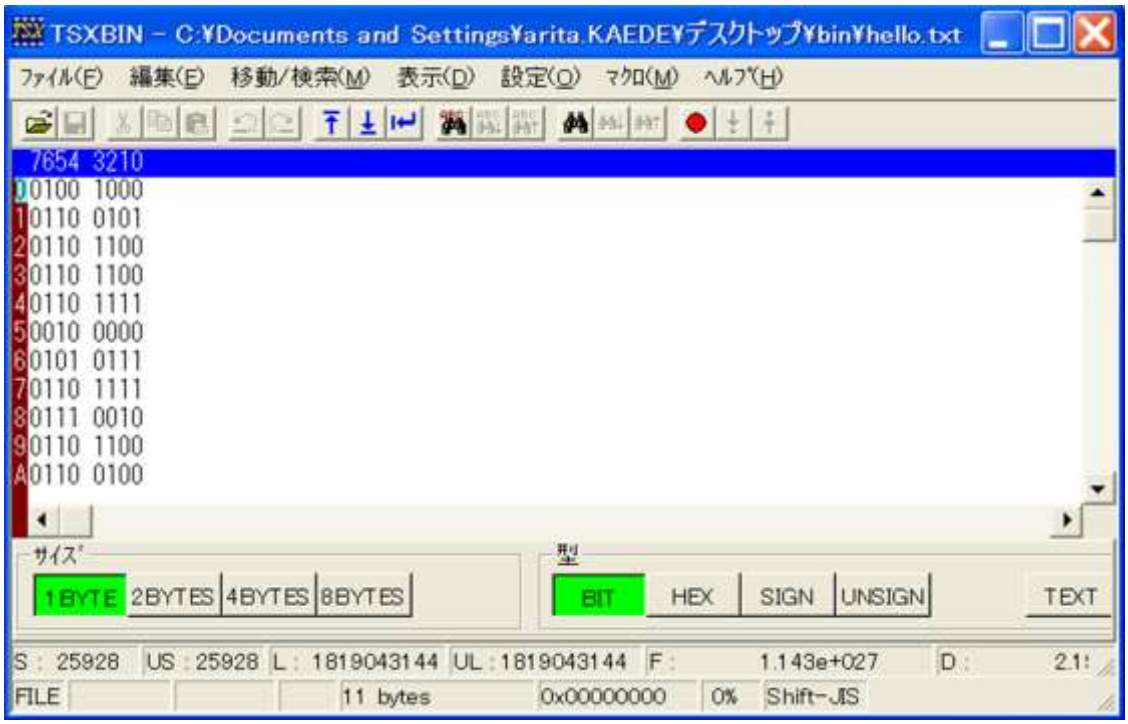
2．ファイルリストに表示する

3．ファイル名を右クリックして
　　プロファイルを選択する

5 バイナリーエディターでファイルを閲覧する

Windows には標準ではバイナリーエディターはインストールされていない。本学にはフリーソフトの TSXBIN というソフトがインストールされている。

1. TSXBIN の起動 （ツールメニュー/バイナリーエディター TSXBIN）
2. TSXBIN のツールバー[ファイル]／[開く]を使用して hello.txt (5 節で作成)を開く。あるいは、hello.txt を TSXBIN の上にドラッグする。
   この状態では Hello Word と ASCII コードとして解釈されて表示されている。
3. TSXBIN の下部にある BIT というボタンをクリックすると、01列として表示されるようになる。



注) 1BYTE 以外のボタンを押すと1行に複数文字分表示されるが並び順が変わってしまう。正しい並びで見たいときは 表示／バイトオーダー／ビックエンディアン を選ぶとよい。