

ファイルをダウンロードしながら、背後でYouTubeを再生し、メールを書くことがなぜ可能なのでしょうか。ファイルのダウンロードで届いたデータがYouTubeの再生データに紛れ込んで表示が乱れることもないし、キーボードを打っている間に動画が途切れることもありません。「マルチタスク」という機能で、2000年ごろにはパソコンでも常識になりました。

OSの目的

リソースの管理

マルチタスクを可能にする。
あるソフトウェアは、同時に別のソフトウェアが実行されていることを心配する必要がない。

基本サービスの提供

文字を表示する、漢字変換する、ファイルに保存する、印刷する機能を、全ソフトが各自用意する必要がない。

ハードウェアの抽象化

どの会社のHD、ディスプレイでも同じように動く。
ハード会社は「ドライバ」だけを提供する

OSの目的は①リソースの管理②基本サービスの提供③ハードウェアの抽象化などです。マルチタスクは、OSが立ち上がっているソフトを400-100分の1秒ごとに自動で切り替えて実行しているので、人間からは同時に動いているように見えているだけです。通信も、ソフトが直接ネットに繋がるのではなく、OSに代行依頼し、OSが自動振り分けをしています。

OSの目的

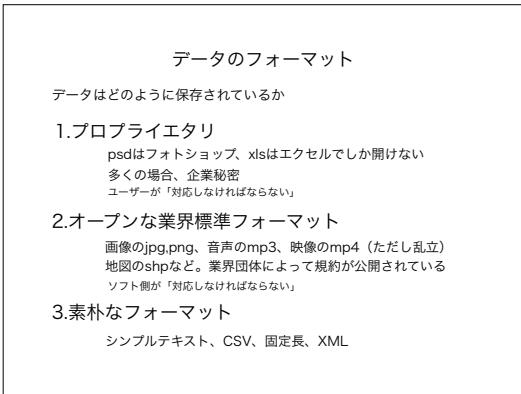
神様の役割

ハングアップしたソフトは、自分自身を終了できないから、OS側から強制終了する必要があります。異なるソフトの間のコピー＆ペーストも、OSがデータを仲介しています。このように、ソフトの多くの機能はOSに依存しています。そのおかげでどのソフトも同じ外観・操作法になりますが、OSが一つの世界を作り上げてしまいます。

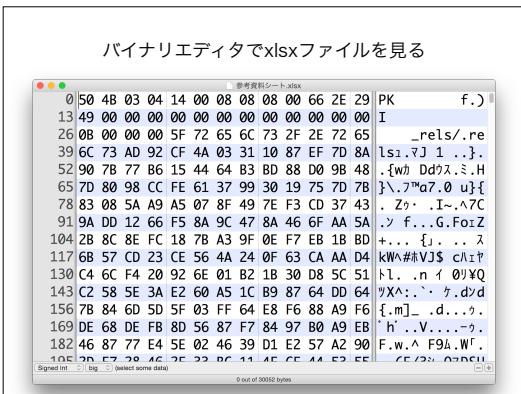
デザインの統一

メニューに印刷、設定、ビューなどがあることなど、共通デザインを半ば強制する。Ctrl+Cなども共通

OSが一つの世界を作り上げてしまう
→データを共有するときに大問題



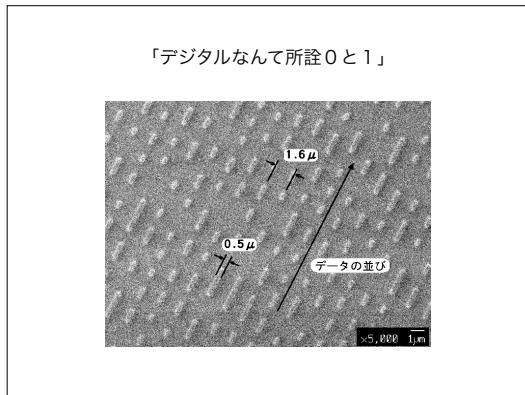
データのフォーマット（形式）は、①psdのフォトショップ、xlsのエクセルのようなプロプライエタリ（企業独自）のフォーマット（多くの場合、レイアウトは企業秘密）②jpegのような業界標準フォーマット（規約は公開）③CSVやXMLなど素朴なフォーマット（規約はあるが、いわば常識）に分かれます。今回説明するのは、素朴なフォーマットです。



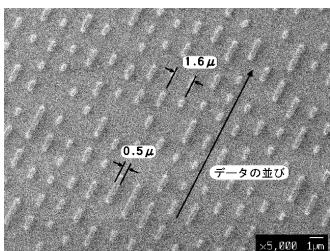
バイナリエディタは、データを直接見ることができるエディタです。WindowsにもMacにも無料のエディタがあります。バイナリエディタは、データを 1 Byte (=8bit) ごとに、16進数で表示するソフトです。図はエクセルシートで、何が書いてあるか、簡単には分かりません。



Jpegの場合も全く分かりませんが、Jpegの規約では、ファイルの最初の方にExifデータ（カメラの機種やシャッター速度、時刻など）が埋め込むことが想定されています。



「デジタルなんて所詮0と1」



写真はCD-ROMの拡大写真です。

デジタルは「所詮ゼロと1」の世界です。ゼロと1の組み合わせで、文字や絵、動画までを表します。問題は、「所詮」と言えるほど、その仕組みを知っているかどうか、です。

IEEE 754

32ビット 単精度 [ソースを編集]

半精度二進浮動小数点数は、32ビットワードに格納される。

sign (exponent (8 bits))

fraction (23 bits)

0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30	23	22	(bit index)																0									

sign は符号、exponent は指部数、fraction は仮数部である。

指部数は下位履歴表現（バイアスまたはエクセプトも。待合付指數表現を参照）と呼ばれる形であります。実際の値に、元の固定値（ここでは $emax = 127$ ）を加算したものです。このよ

うな表現をしているのは浮動小数点範囲の比較をするためである。兩端部が大きな値

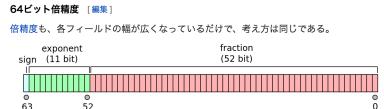
も小さな値を表すように負の値になる。これを単一の正の数と見ると、全体の符号 sign

とは別に exponent も符号を持つことになり、半精度(4ビット)の値が大きくなってしまう。そのため、指部数はバイアスされて正確の正の値になるような形で格納される。単精度では $-1.26 \times +1.27^{127}$ に 2^{127} を加え、 $1 - 2^{53}$ としている（0と255は正の零を表す。後述）。この表現により「数値の正の数」、「数値が0の数」、「0」を、この順に自然と並べることができる。浮動小数点数を操作するときは、バイアスを減して実際の指部数を求める

ことができる。

表現可能なデータは指部数の値によって区別され、仮数部の値にも影響される。指部数も仮数部も符号無しの二進整数であることに注意されたい（指部数は 0~255）。

今は64bitが標準



正規化数では、指数は $emax = +1023$ でバイアスされる（したがって $e = exponent - 1023$ ）。正規化数の指数は $e: +1023 \sim -1022$ ($exponent: 2046 \sim 1$) である ($exponent = 2047$ の時無限大またはNaN、 $exponent = 0$ の時非正規化数で $e = -1022$)。正規化数の時仮数部はけっ切表現である。

コンピューターは「計算機」なので、数字を扱えなければ話になりません。数字はこれまで32bit（32個のゼロと1）で表していました（現在はあまり使われていません）。現在の規約はIEEE754で、23bitで有効数字を表し、8bitで指数を表し、最上位bitで符号を表します。32bitでは日本のGDPを1円まで表せません。有効数字が足りない場合、「丸め誤差」が発生します。現在でも3Dグラフィックスなどで使われています。

現在は64bitが標準

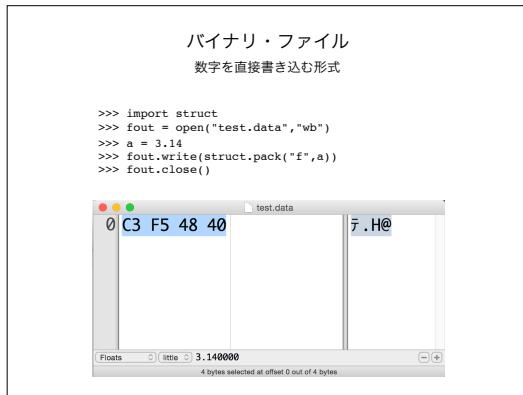
64ビット倍精度 [編集]

倍精度も、各フィールドの幅が広くなっているだけで、考え方は同じである。

field	bits	description
sign	1	sign bit (0 = positive, 1 = negative)
exponent	11	biased exponent (range 0 to 2047)
fraction	52	significand (53 total bits, including the implied leading 1-bit)

正常化数では、指数は $e_{max} = +1023$ でバイアスされる（したがって $e = exponent - 1023$ ）。正常化数の指数は $e = +1023 \sim -1022$ ($exponent: 2046 \sim 1$) である ($exponent = 2047$ の時無限大またはNaN, $exponent = 0$ の時非正規化数で $e = -1022$)。正常化数の時仮数部はけちり表現である。

丸め誤差を避けるため、現在では64bit（倍精度）が標準です。日本のGDPを1銭単位で表現できます。人工衛星の軌道を計算する場合など、128bitが使われる場面もあります。



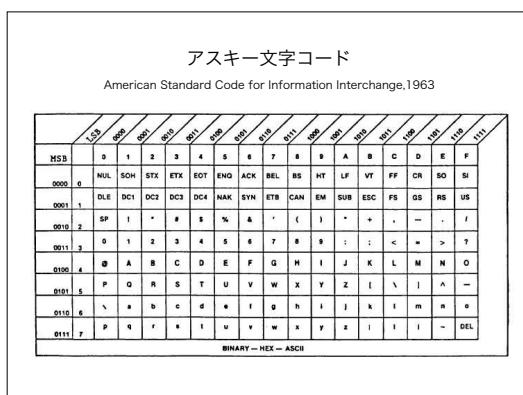
実際に試してみることもできます。図は、Pythonで数字を直接ファイルに保存する例です。

テキストエディタで開くと「テ.h@」のように見えますが、バイナリエディタでは、（ウィンドウの下に表示されているように）3.14であることが確かめられます。



誰が発明したのか分かりませんが、数字を文字に一一対応させれば、数字と同じ仕組みで文字を扱うことができます。計算機をワープロに変えた大発明です。

アメリカ人は「アルファベットと数字、記号を表すには32bitも必要なく、7bitあれば十分」と判断して対応表を作りました。それが「アスキーコード」です。



アスキーワードは128文字(=7bit)です。

Latin1文字コード															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00															
10															
20	!	"	#	\$	%	&	()	*	,	-	.	/		
30	0	1	2	3	4	5	6	7	8	9	:	<	=	?	
40	Ø	A	B	C	D	E	F	G	H	I	J	K	L	M	N
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^
60	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n
70	p	q	r	s	t	u	v	w	x	y	z	{	}	~	
80															
90															
A0	í	ç	ƒ	њ	₩	፣	ؒ	ؑ	ؓ	ؔ	ؕ	ؖ	ؘ	ؙ	ؚ
B0	؎	؏	ؐ	ؑ	ؒ	ؔ	ؕ	ؖ	ؗ	ؘ	ؙ	ؚ	؛	؜	؞
C0	؂	؃	؄	؅	؆	؇	؈	؉	؊	؋	،	؍	؎	؏	ؐ
D0	؎	؏	ؐ	ؑ	ؒ	ؔ	ؕ	ؖ	ؗ	ؘ	ؙ	ؚ	؛	؜	؞
E0	؂	؃	؄	؅	؆	؇	؈	؉	؊	؋	،	؍	؎	؏	ؐ
F0	؎	؏	ؐ	ؑ	ؒ	ؔ	ؕ	ؖ	ؗ	ؘ	ؙ	ؚ	؛	؜	؞

フランス人やドイツ人は「もし8bitにしてくれればアクセントやウムラウトが表示できる」と、独自の文字コードを作りました。Latin1という文字コードで、現在でも欧州で使われています。

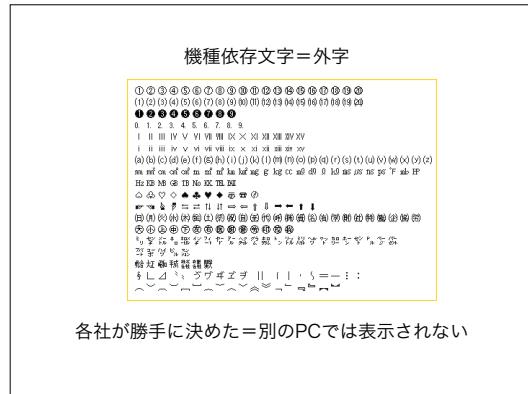
JIS X 0201(半角カナ)															
00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00	N	D	E	S	P	O	@	P	;	—	タ	ミ			
01	SH	D1	!	1	A	Q	a	o	—	—	ア	チ	ム		
02	SX	D2	”	2	B	R	b	r	—	—	イ	ツ	メ		
03	EX	D3	#	3	C	S	c	s	—	—	ウ	テ	モ		
04	ET	D4	\$	4	D	T	d	t	—	—	エ	ト	ヤ		
05	ED	NK	%	5	E	U	e	u	—	—	オ	ヌ	ユ		
06	AK	SN	&	6	F	V	f	v	—	—	カ	ニ	ヨ		
07	BL	EB	*	7	G	W	g	w	—	—	ギ	ヌ	ラ		
08	BS	CN	(8	H	X	h	x	—	—	イ	ク	ネ	リ	
09	H	EM)	9	I	Y	i	y	—	—	ケ	ノ	ル		
0A	LF	SB	*	:	J	Z	j	z	—	—	エ	コ	ハ	レ	
0B	H	MEC	+	;	K	[k	[—	—	オ	サ	ヒ	ロ	
0C	CL	→	,	<	L	¥	l	l	—	—	ヤ	シ	フ	ワ	
0D	CR	←	—	=	M]m	—	—	—	—	ユ	ス	ヘ	ン	
0E	SO	↑	.	>	N	^n	—	—	—	—	ヨ	セ	ホ	。	
0F	SI	↓	?	—	O	—	o	DL	—	—	ツ	ゾ	マ	—	—

日本も「8bitあればカタカナが表示できる」と独自コードを作りました。JIS X 0201です。この頃（1969年）のエンジニアは、コンピューターで漢字を表示することは考えていません。（が、日経ではすでにアネックスシステムの構想がスタートしていました）

漢字の登場																								
256種類では無理→2byteなら256*256まで可能																								
#第1水準漢字(16区～256区)																								
区	点		JIS		SJS		EUC		+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0	3020	8899	BOAO	亞	匪	祉	阿	莫	愛	挨	始	追	英	西	纏	思	提	涅						
16	3030	308AE	B0B0	旭	葦	勝	梓	庄	鈴	拔	羽	姐	虹	鈴	綾	綾	綾	綾	綾	綾	綾	綾	綾	綾
32	3040	88BE	B0C0	粟	袞	安	商	按	確	菴	闇	蕃	杏	以	伊	位	依	偉	圓					
48	3050	88CE	B0D0	夷	委	威	則	惟	意	慰	易	倚	冉	良	異	移	緋	青						
64	3060	88DE	B0E0	萎	衣	謂	進	道	医	井	亥	城	育	都	穢	一	宅	溢						
80	3070	88EE	B0F0	稻	麥	孕	麟	尤	印	姻	眞	因	姻	引	欵	欵	欵	欵	欵	欵	欵	欵	欵	欵
17	3120	893F	B1AO	院	陰	隱	韻	叶	右	宇	烏	卯	近	南	卯	瑞	嘉	丑						
16	3130	894F	B1BO	碓	臼	渴	醴	頤	麌	麌	麌	浦	口	麌	麌	麌	麌	麌	麌	麌	麌	麌	麌	麌
32	3140	895F	B1CO	震	萑	衝	醴	吉	要	影	埃	曳	宋	永	涿	涿	涿	涿	涿	涿	涿	涿	涿	涿
48	3150	896F	B1DO	旗	英	衛	詠	詠	疫	益	駿	駿	駿	駿	駿	駿	駿	駿	駿	駿	駿	駿	駿	駿

漢字は8bit (=256) には収まりません。そこで、1文字を2byte (=256*256) で表すことにし、対応表（文字コード）を作りました。インターネットがなく、コンピューター各社が独自機能を競った時代です。

文字コードの種類	
JIS	JIS X 0208。1978年。第一水準2965字、第二水準3384字を決める。
	メールではISO-2022-JPと呼ばれる
S_JIS	JIS X 0208の亜種。アスキーと半角カタカナを共存させる工夫。Microsoftはcp932、MacはMacJapanese。MS-kanji,x-sjisとも呼ぶ場合も。各社微妙に違う！
EUC-JP	UNIXの世界で、S_JISと同じ目的で作られたコード 文字化けは大抵このレベル



フォントがない												
00	01	02	03	04	05	06	07	08	09	0A	0B	0C
0800	□	□	□	□	□	□	□	□	□	□	□	□
0820	□	□	□	□	□	□	□	□	□	□	□	□
0840	□	□	□	□	□	□	□	□	□	□	□	□
0860	□	□	□	□	□	□	□	□	□	□	□	□
0880	□	□	□	□	□	□	□	□	□	□	□	□
08A0	□	□	□	□	□	□	□	□	□	□	□	□
08C0	□	□	□	□	□	□	□	□	□	□	□	□
08E0	□	□	□	□	□	□	□	□	□	□	□	□
0900	õ	ó	ö	å	ä	å	ä	å	ä	å	ä	å
0920	ç	é	ñ	á	é	ñ	á	é	ñ	á	é	ñ
0940	ñ	é	ó	ú	ñ	é	ó	ú	ñ	é	ó	ú
0960	æ	ø	å	í	ø	å	í	ø	å	í	ø	å
0980	ô	ô	ô	ô	ô	ô	ô	ô	ô	ô	ô	ô
09A0	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö

今まで生き残った文字コードは、JISとシフトJIS、EUCです。JISは日本語専用の文字コードで、シフトJISは、アスキーと半角カタカナを共存させるためにJISをシフトさせた文字コードです。パソコンではシフトJISが使われましたが、Microsoftはcp932、MacはMacJapanese。MS-kanji,x-sjisとも呼び場合もあり、各社微妙に違います。UNIXでもEUCが制定されました。文字化けは大抵このレベルです。

「うちのコンピューターだとこんな表示もできます」と独自の文字も競って導入されました。機種依存文字の誕生です。現在の「絵文字」と同じ現象です。

文字コードが一致していても表示が乱れる場合があります。フォントがない場合です。コンピューターが文字の形状（グリフ）を知らない場合は表示できません。

Unicodeの登場

日本語はS_JIS、簡体字はGB2312(EUC_CN)、繁体字はBIG5/EUC_TW、韓国語はJohab/UHCで共存できない。フォントも違う。
→ 対訳テキストが作れない！

全ての言語の文字を表す文字コード=Unicode

4byteで文字を表現する=4byte使うとは限らない
UTF-8: アスキーは1byte、よく使う字は2byte、稀に使う時は3,4byte

UTF-16: 2byteか4byte
UTF-32: 常に4byte
公的文書も最近はUTF8になりつつあります

改行コード問題

LF(10): Line Feed=行送り
CR(13): Carriage Return=印刷位置の復帰

Windows	LF+CR
iMac	CR
Unix, Mac OS	LF

Macで作ったtext/csvファイルを読み込んだ
Windowsは、いつまでたってもCRが来ない！
ただし、賢いソフトは補ってくれる

テキスト・ファイル

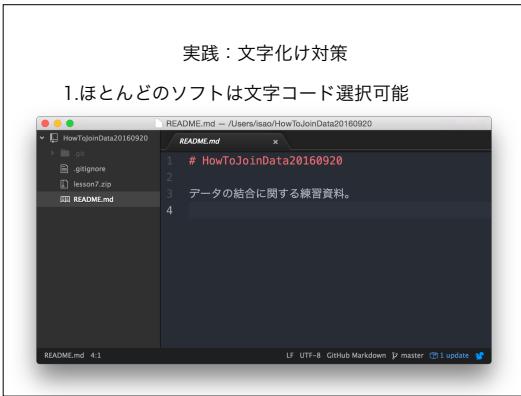
abc<改行>あ<改行>

abc: utf8とshiftJISは同じ、utf16は00がついている
改行: utf8は0A、shiftJISは0D0A（Macは0D）、utf16は0A00
あ: utf8はE38182、shiftJISは82A0、utf16は3042(LEは順序反転)

インターネットが普及すると問題が露わになりました。日本語はS_JIS、簡体字はGB2312(EUC_CN)、繁体字はBIG5/EUC_TW、韓国語はJohab/UHCで、コードも違えば、フォントも違うため、対訳テキストが作れません。いまでもS_JISを使っているサイトは金妍兒が表示できません（ガラケーがS_JISだからです）。そこで、全ての言語の文字を表す文字コード=Unicodeが制定されました。最近の公的文書はUTF8になりつつあります。

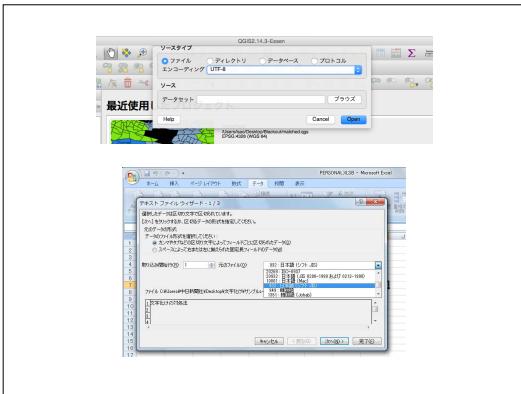
文字コードと違い、改行コードの問題もあります。改行を表す記号がWindowsと旧Mac、Unix（現在のMacもUnixの一種）で違います。WindowsのソフトはLF+CRの2文字を改行と解釈するので、もしUnixで作られたファイル（改行をLFだけで表す）を読み込むと、改行が一つもないと判断します。MacのQGISで作ったファイルをWindowsのQGISで読み込むと失敗します。

改行の様子を実際に見てみましょう。図は、「abc<改行>あ<改行>」を入力し、shift-jisとUTF8、UTF16で保存し、バイナリーエディタで表示したものです。改行部分は、utf8は0A、shiftJISは0D0A、utf16は0A00になっています。



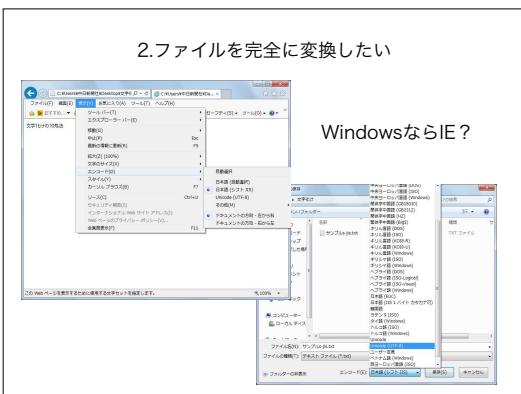
実際に文字化けにどう対処すべきでしょうか。

ほとんどのソフトは文字コード選択可能。エディタには、ファイルを開く際に文字コードを選択できるものや、メニューで文字コードを変更できるものがあります。



QGISには「エンコーディング」を選択できるオプションがありました。

Excelにも、テキストファイル（CSVなど）を読み込むダイアログに文字コードを選択できるオプションがあります。



ファイルを丸ごと変換したい場合、WindowsならIEで開いてエンコードを選び直して表示を確認し、ファイルを保存する場合にエンコードを指定することができます。

2. ファイルを完全に変換したい

Mac/Ubuntuならコマンドライン

```
nkfは富士通製変換コマンド（超機能！）
> nkf -guess sample.txt
UTF-8
> nkf -s sample.txt > s-jis.txt // 文字コード
> nkf -Lw sample.txt > CRLF.txt // 改行形式

iconvは世界共通（改行形式はそのまま）
> iconv -f SHIFT-JIS -t UTF8 sample.txt > utf8.txt
```

腹をくくってコマンドラインを習得しよう

```
> nkf -w -Lu -overwrite *.txt
あるフォルダのtxtファイルを一括変換
```

文字コードが違う文書で
コピペはなぜ可能？

CSVファイル

#	地震データ
1	"time","day","lat","lon","mag","bsho"
2	"16/04/20 13:47","20,33,131.11,1.4,"阿蘇山周辺"
3	"16/04/20 13:45","20,32,79,130.95,1.6,"阿蘇山周辺"
4	"16/04/20 13:42","20,33,04,131.14,1.4,"熊本県北東部・大分県境付近"
5	"16/04/20 13:31","20,32,66,130.67,2.6,"熊本県中部"
6	"16/04/20 13:27","20,32,62,130.68,0.9,"熊本県中部"
7	"16/04/20 13:27","20,32,55,130.55,1,"八代海・熊本県南西部"
8	"16/04/20 13:26","20,32,19,130.51,1.8,"八代海・熊本県南西部"
9	"16/04/20 13:26","20,32,19,130.49,1.5,"八代海・熊本県東南部"
10	"16/04/20 13:22","20,32,92,130.99,2.2,"阿蘇山周辺"
11	"16/04/20 13:14","20,32,57,131.35,2.6,"宮崎県北部"
12	"16/04/20 13:07","20,32,59,130.35,1.1,"熊本県中部"
13	"16/04/20 13:04","20,33,28,131.41,1.9,"別府付近"
14	"16/04/20 13:03","20,32,87,130.84,1.7,"熊本県北部"
15	"16/04/20 12:47","20,32,96,131.1,2.4,"阿蘇山周辺"
16	"16/04/20 12:38","20,32,89,130.91,1.4,"阿蘇山周辺"
17	"16/04/20 12:38","20,32,83,130.85,1.5,"熊本県北部"
18	"16/04/20 12:34","20,32,97,131.12,1.3,"阿蘇山周辺"
19	"16/04/20 12:33","20,32,71,130.68,3.2,"熊本県中部"
20	"16/04/20 12:31","20,32,92,131.03,1.7,"阿蘇山周辺"
21	"16/04/20 12:26","20,32,51,130.62,0.9,"熊本県中部"
22	"16/04/20 12:21","20,33,05,131.14,1.2,"熊本県北東部・大分県境付近"

MacにはIEがありません。Safariは表示までは文字コード変換できますが、保存ができません。腹をくくってコマンドラインを使いましょう。

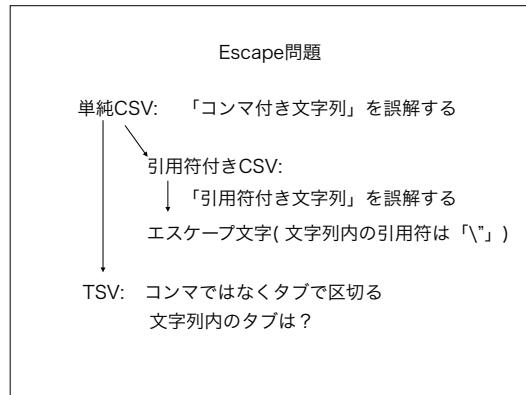
iconvというコマンドは標準装備です。ただし、文字コードは変換できますが、改行形式はそのままになります。改行も変更したい場合は、nkfというコマンドをインストールする必要があります。（インストール方法はネットで見つかります）

ところで、文字コードが違う文書でコピペはなぜ可能なのでしょうか？

コピーは、OSのクリップボードにデータを転送することです。クリップボードには型情報付きのデータを何種類でも転送でき、ペーストする時に都合のいい型を選んで取得できます。表計算ソフトに「形式を選択してペースト」があるのはこの機能です。通常のテキストエディタは、UTF32に変換してコピーし、ペーストするときはUTF32から変換して貼り付けます。UTF32はOSの内部で広く使われている文字コード（いわば共通語）です。

素朴なフォーマットについて説明します。

CSVファイルはComma-Separated Valuesです。文字列は二重引用符で包み、数字は直接書かれています。ただし、例外多数。引用符がなかったり、シングル引用符だったり、コンマの代わりにセミコロンだったりする場合もあります。1行目は列名の場合もあれば、列名がない場合もあります。



問題は3つ。①型情報は保存されない（整数なのか、日付データなのか分からぬ）②改行はOSによって違う③分割文字（コンマ）を表現するために「エスケープ文字」が必要になります。エスケープ文字はプログラムを書く場合にも必要な基本知識です。（ただし、言語・ソフトによって微妙に異なる）

固定長データ											
測定ID	測定名	測定値	測定値	測定値	測定値	測定値	測定値	測定値	測定値	測定値	測定値
J_01_S00240001B4B	RES 362699	017	1374127	017	21031210V	511	4140HIDA MOUNTAINS REGION				20K
N_0MCH2748	K4P	000136655	0174347 1762 01	1 1139 01	1 1181 08	01					15 GAM 1
E_MTU	B8T 1247	000137955	013978 607 02	3 627 03	3 627 05	03	6551 23				15 GWH 1
N_0MCH2748	K4P	000138245	014018 1193 03	1 1139 01	1 1181 08	01	6854 23				15 GAM 1
N_0MCH2748	K4P	000138245	014018 2283 02	2 1279 00	2 1280 04	00	6854 23				15 GAM 1
N_0MMH2796	K4P	00013983	02830 01 2 3014 03	5 0181 03	5 0181 03	03					15 GM 1
DPT	TP131367	1245		221 00	4 24 00	04	221 02 03				15 GM 1
N_0IKSH2674	K4P	00014045									15 GM 1
N_0SBRS2705	K4P	000140453									15 GM 1
N_0MSTH2515	K4P	000140455	014388 153 00 04	1 101 03	1 101 04	03	3552 43				15 GMW 1
N_0IKSH2674	K4P	000140455	014378 119 01 04	1 101 03	1 101 04	04	8810 43				15 GAM 1
N_0HAB2749	K4P	00014442	014553 160 00 04	1 101 03	1 101 04	04	8810 43				15 GM 1
N_0HAB2749	K4P	00014442	014553 160 00 04	1 101 03	1 101 04	04	8810 43				15 GM 1
N_0SKA2749	K4P	00014215	014699 160 00 02	1 143 01	1 143 01	02	4741 63				15 GAM 1
DPT	TK131324	1245	00014670	183 01 00	2 1357 00	00	140 01 03				15 GM 1
E_AER	480 1245	00014732	014726 120 01 00	1 101 03	1 101 04	00	172 08 53				15 GM 1
N_MATUS	67 1249	00014835	014985 106 01 00	1 106 01	1 106 01	00	56 01 7	56 01 7	84 02 73		15 GAM 1
E_NMM2748	K4P	000148351	014961 106 01 00	1 106 01	1 106 01	00	56 01 7	56 01 7	84 02 73		15 GAM 1
Z_E0528524008341A3	013 35285 041	14113676	078 349232682	511	3110NOSHIO CHOSHI CITY						28K
TENND	699	000349885	003554 444 00	4 3579 00	4 3579 00	00	5668 01 13				15 GMW 1
N_CHS2748	K4P	000349875	003579 1818 03	7 1809 04	9 2669 02	13	5668 01 13				15 GMW 1
N_HAH2746	K4P	000356885	003608 120 01 00	1 101 03	1 101 04	00	5668 01 13				15 GMW 1
N_HAH2746	K4P	000356885	003608 120 01 00	1 101 03	1 101 04	00	5668 01 13				15 GMW 1
N_HAH2746	K4P	000356885	003608 120 01 00	1 101 03	1 101 04	00	5668 01 13				15 GMW 1
N_NSM2799	K4P	000352465	004045 1321 05 00	16 1423 04	14 1353 02	83					15 GMW 1
N_NSM2799	K4P	00035331	004045 1321 05 00	16 1423 04	14 1353 02	83					15 GMW 1

固定長データは、最もデータ密度が高い形式で、メモリーが貴重だった頃から使われています。現在でも、通信容量が貴重なGPSや人工衛星データで使われています。

何文字目が何を表すか決まっているので、プログラムで読み取ることが容易。表計算ソフトやopenRefineには、固定長データの切り分け機能があります。

固定長データは「レコードレイアウト」があって初めて解釈できます。図の場合、型情報は、I2は整数2桁、F4.2は4桁で下2桁が小数点以下であることを意味しています。マグニチュードで「A2」は-0.2を意味するなど、独自のコード化がされている場合もあります。だから、データのコーディング規約は必読です（参考：2016/9/25毎日朝刊の訂正）

XMLデータ

HTML（XMLの一種）のように、データをタグで記述する方式

要素数が事前に決められない場合
データの定義が固定できない場合
データの検証が必要な場合
データ交換の業界標準方式
ただし、データサイズは巨大

```

<bookstore>
  <book isbn="978-4-7741-5715-3"> <!-- 属性(attribute)として
    <title>本のタイトル</title> <!-- 必ず必要な要素として
    <author>著者の名前1 </author> <!-- 著者は1人以上
    <author>著者の名前2 </author>
  </book>
</bookstore>

```

WNPAデータ（国別団体）

```

1 <?xml version="1.0" encoding="UTF-8"?><OfferingXML Version="5.0">
2   <Header Category="5" CreateDate="2014/02/09" CreateTime="15:11:00" DeliveryXmlVersion="1"
3     ReportNumber="00003371" ResultSystemCode="FSX400000" XmlVersion="0"/>
4   <Meta CategoryTitle="オフィシャルリザルト" SubTitle="団体属性" Title="フィギュア" TransmittDate="2014/02/10 02:58:39"/>
5   <!-- リザルト登録情報 -->
6   <ResultList>
7     <ParticipantUnit CountryName="ロシア" IOCCode="RUS" MemberCount="15" Rank="1">
8       <Member Type="Point" Value="75"/>
9       <Member>
10         <Athlete FamilyName="プリュ申コ" FamilyNameEntry="Plyushchenko" FormatType="4"
11           GivenName="エフゲニー" GivenNameEntry="Evgeny" Id="2001933" TName="Evgeny
12             PLYUSHCHENKO"/>
13         <ParticipantInfo Type="MemberNumber" Value="1"/>
14       </Member>
15     </Member>
16     <Member>
17       <Athlete FamilyName="ソトニコワ" FamilyNameEntry="Sotnikova" FormatType="4"
18         GivenName="アデリナ" GivenNameEntry="Adelina" Id="2001886" TName="Adelina SOTNIKOVA"/>
19       <ParticipantInfo Type="MemberNumber" Value="2"/>
20     </Member>
21     <Member>
22       <Athlete FamilyName="リノツカヤ" FamilyNameEntry="Lipnitskaya" FormatType="4"
23         GivenName="ユリヤ" GivenNameEntry="Yuliya" Id="2001879" TName="Yulia LIPNITSKAYA"/>
24       <ParticipantInfo Type="MemberNumber" Value="3"/>
25     </Member>
26   </ResultList>
27   <ReportNumber>00003372</ReportNumber>
28   <ParticipantInfo Type="MemberNumber" Value="4"/>
29 </Header>
30 <Member>
31   <Athlete FamilyName="ボロゾハル" FamilyNameEntry="Voloszhar" FormatType="4"
32     GivenName="タチアナ" GivenNameEntry="Tatiana" Id="2001872" TName="Tatiana VOLOSZHAR"/>
33   <ParticipantInfo Type="MemberNumber" Value="5"/>
34 </Member>
35 <Member>
36   <Athlete FamilyName="トランコフ" FamilyNameEntry="Trankov" FormatType="4"
37     GivenName="マキシム" GivenNameEntry="Maxim" Id="2001943" TName="Maxim TRANKOV"/>
38 </Member>

```

WNPA個別データ

```

1 <?xml version="1.0" encoding="UTF-8"?><OfferingXML Version="5.0">
2   <Header Category="5" CreateDate="2014/02/09" CreateTime="15:11:00" DeliveryXmlVersion="1"
3     ReportNumber="00003372" ResultSystemCode="FSX400000" XmlVersion="0"/>
4   <!-- リザルト登録情報 -->
5   <ResultList EventDay="2014/02/09" EventTime="10:00">
6     <ParticipantUnit CountryName="イタリア" IOCCode="ITA" MemberCount="1" StartingNumber="1">
7       <Member>
8         <Athlete FamilyNameEntry="Parkinson" GivenNameEntry="Paul Bonifacio" Id="2015464"
9           TName="Paul Bonifacio PARKINSON"/>
10        <ExerciseInfo Code="45" Description="Quad Salchow" Order="1"/>
11        <ExerciseInfo Code="34" Description="Double Axel+Loop" Order="2"/>
12        <ExerciseInfo Code="3A" Description="Triple Axel" Order="3"/>
13        <ExerciseInfo Code="CSp" Description="Change Foot Combination Spin" Order="4"/>
14        <ExerciseInfo Code="FSOp" Description="Flying Camel" Order="5"/>
15        <ExerciseInfo Code="3Q" Description="Triple Loop" Order="6"/>
16        <ExerciseInfo Code="3Lz+3T" Description="Triple Lutz+Triple Toeloop" Order="7"/>
17        <ExerciseInfo Code="2x1Lo+3S" Description="Double Axel+Single Loop+Triple Salchow"
18          Order="8"/>
19        <ExerciseInfo Code="3F" Description="Triple Flip" Order="9"/>
20        <ExerciseInfo Code="3Lo" Description="Triple Loop" Order="10"/>
21        <ExerciseInfo Code="3T" Description="Triple Toe loop" Order="11"/>
22        <ExerciseInfo Code="3Lz" Description="Triple Lutz" Order="12"/>
23        <ExerciseInfo Code="CSOp" Description="Change Foot Sit Spin" Order="13"/>
24      </ParticipantUnit>
25      <ParticipantUnit CountryName="米国" IOCCode="USA" MemberCount="1" StartingNumber="2">
26        <Member>
27          <Athlete FamilyName="ブロウ" FamilyNameEntry="Brown" FormatType="4" GivenName="ジョン"
28            GivenNameEntry="Jason" Id="2036672" TName="Jason BROWN"/>
29        </Member>
30        <ExerciseInfo Code="2A" Description="Double Axel" Order="1"/>
31        <ExerciseInfo Code="3A+3T" Description="Triple Axel+Triple Toeop" Order="2"/>

```

XML形式は、①本の著者のように要素数が決まっていない場合、②毎年ルールが変わる競技のように定義が固定できない場合、③isbn番号のように欠けてはいけない項目があるなど検証が必要な場合などに使います。現在ではデータ交換の業界標準方式ですが、データサイズは巨大になります。

これは、五輪の公式データ（WNPA）の一部です。このデータを手作業で編集するには「XMLエディタ」を使いますが、基本的には、XMLはプログラムで読むためのものです。主要なプログラミング言語には「XMLパーサー」という部品が用意されています。

フィギュアスケートのデータです。数が事前に確定しない競技要素（エレメント）がExerciseInfoというタグで記述されています。

このように、複雑なデータはXMLで記録・配信されます。データを読み取るにはプログラミングが不可欠です。腹をくくりましょう。表計算のシートで表示できるデータは現実をかなり抽象化したデータです。

ちなみに、国際スケート連盟は、オリンピックでも競技終了後数分でPDFの採点表を独自サイト（www.isu.org）に公開します。綺麗なPDFなので、ダウンロードしてExcel変換サービスで変換したほうが加工は簡単です。フィギュアは（採点スキャンダル後の改革によって）採点が高度にシステム化されているため、このような情報提供が可能です。

これらの基本的な知識は、大学の1年生向け公開資料などを探すとまとめて勉強できます。

參考資料

大学の1年生向け情報処理の講義
[http://www.coins.tsukuba.ac.jp/~yas/coins/
literacy-2016/2016-05-06/](http://www.coins.tsukuba.ac.jp/~yas/coins/literacy-2016/)