TIS
TIS INTEC Group

文字の扱い

TIS株式会社
テクノロジー＆イノベーション本部
テクノロジー＆エンジニアリングセンター

商標

- Microsoft WindowsはMicrosoft Corporationの商標または登録商標です。
  なお、本文中ではWindowsと表記しています。

その他の社名、製品名などは、一般にそれぞれの会社の商標または登録商標です。
なお、本文中では、TMマーク、Rマークは明記しておりません。

目次

- はじめに
- 文字の基礎知識
  - 文字コードと文字集合
  - Unicodeとその符号化方式
- 業務での応用方法
  - プロジェクトにおける文字関連のタスク

付録 Shift_JIS、EUC-JP、ISO-2022-JP

はじめに ～基礎用語～

文字コード

コンピュータ上で文字（キャラクタ）を利用する目的で各文字に割り当てられるバイト表現。もしくは、バイト表現と文字の対応関係（文字コード体系）のこと。

出典：フリー百科事典　ウィキペディア日本語版

https://ja.wikipedia.org/wiki/%E6%96%87%E5%AD%97%E3%82%B3%E3%83%BC%E3%83%89
2021年12月7日14時の最新情報を取得

文字コード

文字をビット表現に対応付けた例

文字

文字に対応したビット表現

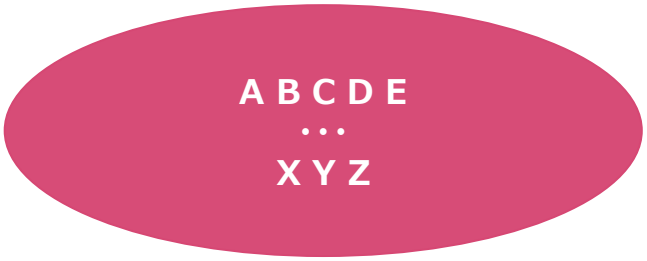
A = 0001
B = 0010
C = 0011
:

1ビットなら2文字、2ビットなら4文字、…、
nビットなら2^nの文字を表現することが可能

文字集合

コンピュータ上で文字を扱う場合、どのような文字集合を使うか、あらかじめ取り決めておく必要がある。

平仮名



アルファベット

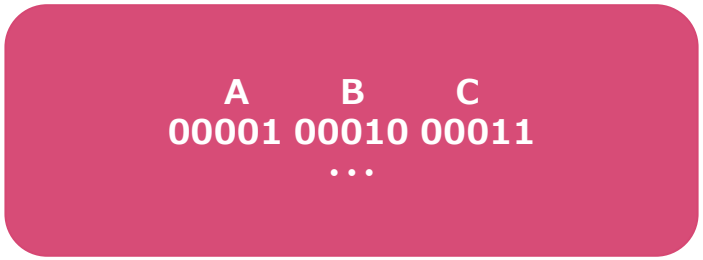
数字

常用漢字

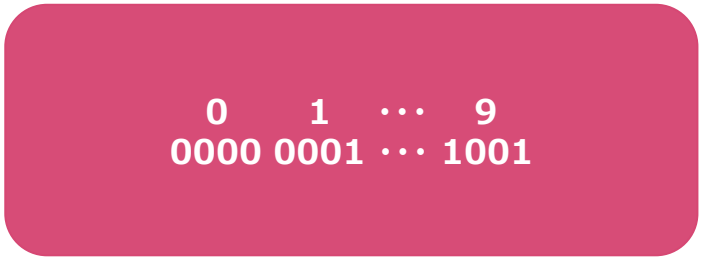
符号化文字集合

文字集合を定義して、その集合内の各文字に一意の
符号化表現を関連付ける規則を符号化文字集合と言う。

平仮名



アルファベット



数字

常用漢字

代表的な符号化文字集合

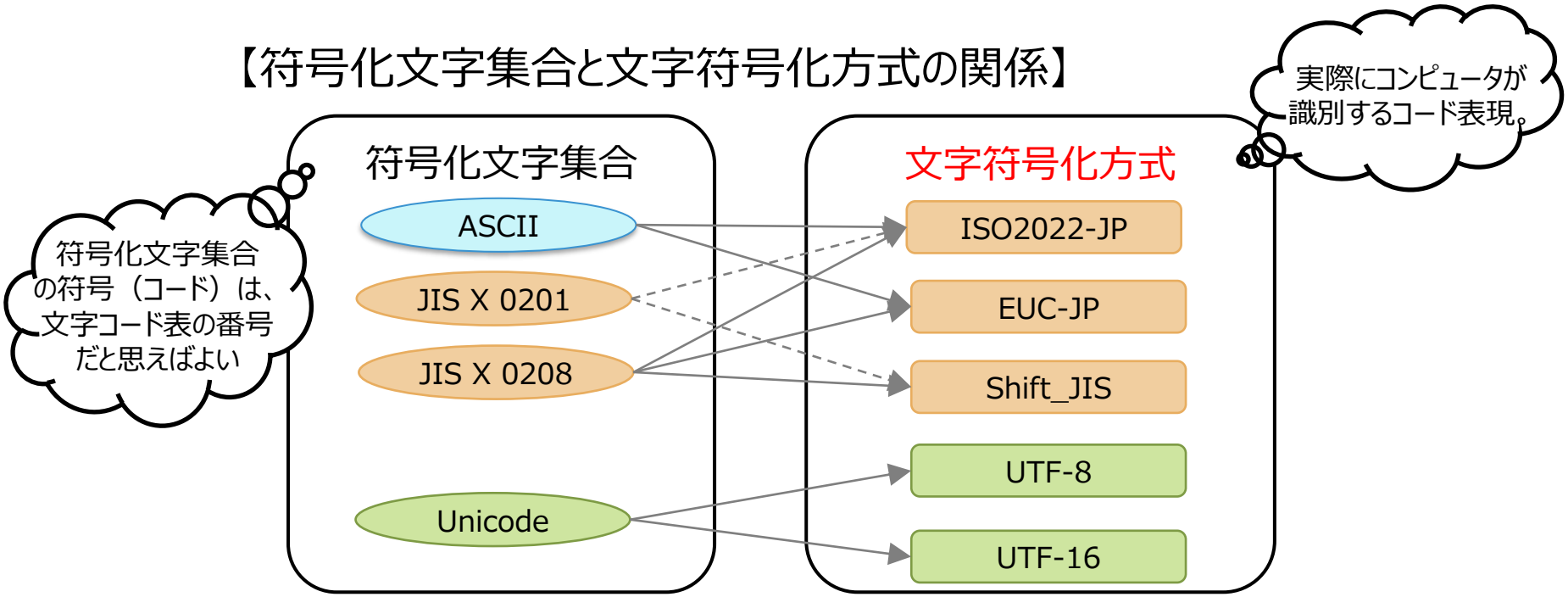
規格	説明
ASCII	米国のANSI規格。最も基本的な文字コード。
ISO/IEC 8859-1	ASCIIに加えて、アクセント記号付きアルファベット等を収録した西ヨーロッパ向けの規格。通称、Latin-1。
JIS X 0201	日本規格。ラテン文字用図形文字集合と片仮名用図形文字集合のふたつの文字集合よりなっている。
JIS X 0208	日本の漢字、平仮名、片仮名等を収録。 1978年初版。第1・第2水準漢字を含む6,879文字。
JIS X 0213	JIS X 0208に足りない文字を追加した拡張版。 2000年初版。第3・第4水準漢字を含む11,233文字。
Unicode	世界中の文字を単一の符号化文字集合でカバーすることを目的とした規格。絵文字や記号を含む世界各言語の143,859文字を収録。

文字符号化方式

ASCIIやISO/IEC8859のような符号化文字集合は、それ単体で運用されることが多いが、EUC-JPやShift_JISなどは2つ以上の符号化文字集合を組みわせたり変形したりして運用される文字コードである。

符号化文字集合を運用しやすいように別のバイト列に変換する方式を文字符号化方式と言う。

【符号化文字集合と文字符号化方式の関係】



主な符号化文字集合と文字符号化方式

符号化文字集合	文字符号化方式	説明
ASCII		米国のANSI規格。最も基本的な文字コード。
JIS X 0201		日本規格。ラテン文字と片仮名のふたつの文字集合を含む。
ISO/IEC 8859-1		ASCIIに加えて、アクセント記号付きアルファベット等を収録した西ヨーロッパ向けの規格。通称、Latin-1。
JIS X 0208		日本の漢字、平仮名、片仮名等を収録。第1・第2水準漢字を含む6,879文字。
	ISO-2022-JP	ASCII、JIS X 0201ラテン文字、JIS X 0208文字集合を扱う。電子メールで長年使用された符号化方式。
	EUC-JP	ASCIIとJIS X 0208文字集合を扱う。Unix上で利用されてきた符号化方式。
	Shift_JIS	JIS X 0201とJIS X 0208文字集合を扱う。Windows系システムや携帯電話で使用されてきた符号化方式。
JIS X 0213		JIS X 0208に足りない文字を追加した拡張版。 2000年初版。第3・第4水準漢字を含む11,233文字。
Unicode		世界中の文字を単一の符号化文字集合でカバーすることを目的とした規格。世界各言語の143,859文字を収録。
	UTF-8	Unicodeで一番利用される形式。1～4バイトの可変長。
	UTF-16	Unicodeを16ビットで表現。ただし、サロゲートペアは32ビット。
	UTF-32	Unicodeを32ビットで表現。

本講座で学ぶこと

本講座では文字コードに関する以下の基礎知識を
学習します。

- 日本で使われている主要な文字集合
- 現在の主流となるUnicodeとその符号化方式

そのうえで、プロジェクトにおける文字関連のタスクを確認して
いきます。

文字コードと文字集合

ASCII

最初にコンピューターが発達したアメリカで英語を書くために必要な文字の符号化が試みられ、ASCIIが作られた。

- 1963年6月17日、米国国家規格協会（ANSI）が制定
- 1バイトのうちの7ビットを利用して1文字を表す文字コード。
  128の符号位置があり、以下が割り当てられている。

  制御文字（control character）
    - 画面に表示するための文字ではなく、モニタやプリンタなどの機器を制御するために用いられる文字。

  図形文字（graphic character）
    - 画面に表示できる一般的な文字。

  ※符号位置は、文字集合内の文字を割り当てうる個々の点。

ASCIIの文字コード表

上位3ビット →

下位4ビット ↓

b4	b3	b2	b1	0x	b7=0 b6=0 b5=0	b7=0 b6=0 b5=1	b7=0 b6=1 b5=0	b7=0 b6=1 b5=1	b7=1 b6=0 b5=0	b7=1 b6=0 b5=1	b7=1 b6=1 b5=0	b7=1 b6=1 b5=1
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

16進数表記

制御文字の符号位置

図形文字の符号位置

例えば、「z」のビット表現は1111010。16進数表記では0x7Aで表される。

ASCIIの国際規格化

ASCIIをそのまま使うのでは、アメリカ以外の国では不都合がある。そのため、ASCIIの一部の文字や記号を別のものに取り換えて各国用の文字コードを作る枠組みが策定された。

ISO/IEC646は、ASCIIをベースにして各国の都合に応じて文字を変更してよい符号位置**12文字**を定めている。

ISO：国際標準化機構

IEC：国際電気標準会議

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1	0x	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

日本での標準化

JIS X 0201は、ISO/IEC646に従って日本工業規格が標準化した規格。変更可能な12文字のうち2文字を変更した。

SP	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(	8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[	k	{
,	<	L	\	l	|
-	=	M	]	m	}
.	>	N	^	n	~
/	?	O	_	o	DEL

SP	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(	8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[	k	{
,	<	L	¥	l	|
-	=	M	]	m	}
.	>	N	^	n	‾
/	?	O	_	o	DEL

バックスラッシュ→円記号
符号位置：0x5C

チルダ→オーバーライン
符号位置：0x7E

JIS X 0201

2つの1バイト集合を持つ符号化文字集合。

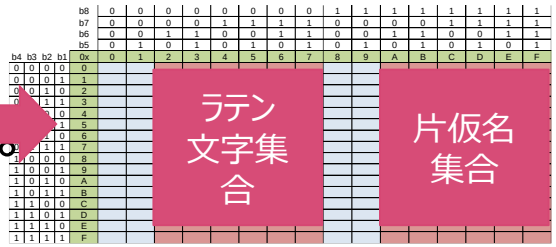
ラテン文字集合

- ASCIIと2文字異なる
  （バックスラッシュ→円記号、チルダ→オーバーライン）

片仮名集合

- 濁点、半濁点のついた文字は含まれておらず、
  濁点（0x5E）と半濁点（0x5F）は独立した文字として用意。
- 0x60～0x7Fは未定義

- 7ビットの運用方式と8ビットの運用方式がある
  - 7ビット符号：制御文字（SHIFT-OUT、SHIFT-IN）によって、ラテン文字集合と片仮名集合を切り替える。
  - 8ビット符号：8ビットを利用することで、
    7ビット範囲の符号が同じコードが2面できる。



JIS X 0208

JIS X 0208は、日本で使われる漢字・ひらがな・カタカナなどを収めた2バイトの符号化文字集合。

- 理論上、8,836文字の収録が可能。
  - 最新版の1997年版には　6,879文字
    - JIS第1水準漢字　：2,965文字
    - JIS第2水準漢字　：3,390文字
    - 非漢字　：524文字
      （特殊文字、数字、ラテン文字、平仮名、片仮名など）
- 文字コード表は94行×94列で表される。

2バイトを使用することで表現できる文字を拡張

JIS X 0208 の文字コード表

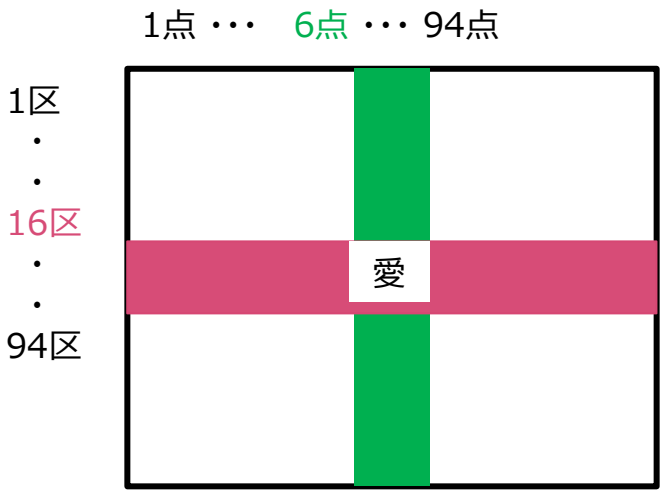
第1・第2バイトそれぞれ0x21～0x7Eの94種類の組合せ。
→94行（第1バイト）×94列（第2バイト）の文字コード表

文字コード表は、**区点番号**を使用して94行×94列の文字表の中の位置を示す構成となっている。

- 区番号：文字コード表の行
- 点番号：文字コード表の列

例）

愛	第1バイト	第2バイト
16-06	0110000	0100110



JIS X 0208 の収録文字

区番号	文字
1区・2区	特殊文字（句点・読点、括弧、通貨、記号など）
3区	数字、ラテン文字
4区	ひらがな
5区	カタカナ
6区	ギリシャ文字
7区	キリル文字
8区	罫線素片（─、┐、├など）
9区～15区	未定義
16区～47区	JIS第一水準漢字
48区～84区	JIS第二水準漢字
85区～94区	未定義

収録漢字の水準分け

第1水準漢字

- 当用漢字字体表、当用漢字補正案および人名用漢字別表を基本として、多種の漢字表に共通して出現する文字が選ばれた。
- また、都道府県名および市区町村名に使用される漢字がすべて第1水準に含まれるように意図された。

第2水準漢字

- 主要な漢字表に出現して第1水準から漏れた漢字が収められた。

なお、現在は**JIS X 0213**という上位互換規格があり、
第3水準漢字、第4水準漢字が定義されている。

JIS X 0208、JIS X 0213の収録文字数

TIS INTEC Group

規格	通称	制定年	第1水準漢字	第2水準漢字	第3水準漢字	第4水準漢字	非漢字	合計
JIS C 6226 ※	78JIS	1978年	2,965字	3,384字	-	-	453字	6,802字
JIS X 0208	83JIS	1983年		3,388字			524字	6,877字
	90JIS	1990年		3,390字				6,879字
	97JIS	1997年						
JIS X 0213	JIS2000	2000年			1,249字	2,436字	1,183字	11,223字
	JIS2004	2004年			1,259字			11,233字

※JIS C 6226 は、JIS X 0208 の旧規格番号。

https://www.tohoho-web.com/ex/charset.html
2021年12月7日14時の最新情報をもとに作成

Shift_JIS、EUC-JP、ISO-2022-JP

元々、ASCII や JIS X 0201 で運用されていたシステムでは、既存のプログラムはラテン文字や数字等を1バイトで表現することが前提となっていた。

そうした資産を2バイトコードに移行することは現実的ではないため、JIS X 0208 を ASCII やJIS X 0201 といった1バイトコードと組み合わせて運用する方式が開発され、広く普及した。

よく知られる**Shift_JIS**や**EUC-JP**、**ISO-2022-JP**は、そうした文字符号化方式である。

（Shift_JIS、EUC-JP、ISO-2022-JPについては付録を参照。）

ヨーロッパでの発展

ISO/IEC8859（シリーズ）は、8ビットの1バイトコードで、ヨーロッパ内のまとまった**地域**で使われる文字を文字集合に収めて扱う規格。

パート	地域・言語	パート	地域・言語
第1部	**ラテン1 西ヨーロッパ**	第9部	ラテン5 トルコ語
第2部	ラテン2 中央ヨーロッパ	第10部	ラテン6 北ゲルマン語群
第3部	ラテン3 南ヨーロッパ	第11部	ラテン/タイ
第4部	ラテン4 北ヨーロッパ	第12部	ラテン/デーヴァナーガリー
第5部	ラテン/キリル	第13部	ラテン7 バルト語
第6部	ラテン/アラビア	第14部	ラテン8 ケルト語
第7部	ラテン/ギリシア	第15部	ラテン9
第8部	ラテン/ヘブライ	第16部	ラテン10 南東ヨーロッパ

第12部：1997年に破棄

ISO/IEC 8859-1（第1部パート）

ISO/IEC 8859-1は、ほとんどの西ヨーロッパ言語をカバーし、もっとも広く使われているパートである。通称**Latin-1**。

- 完全に網羅している言語
  - アフリカーンス語、アルバニア語、ブルトン語、デンマーク語、英語、フェロー語、ガリシア語、ドイツ語、アイスランド語、アイルランド語、イタリア語、ラテン語、ルクセンブルク語、ノルウェー語、オック語、ポルトガル語、レト・ロマンス語、スコットランド・ゲール語、スペイン語、スワヒリ語、スウェーデン語、ワロン語、日本語　（訓令式ローマ字）
- ほぼ網羅している言語
  - オランダ語、エストニア語、フランス語、フィンランド語

ISO/IEC 8859-1 （Latin-1）

8ビットを利用するためASCIIと同じ構造が2面もてる。

左側（CL、GL領域）はASCIIと同じ。右側のGR領域にその地域で必要な文字を収録する。

				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1	0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0																
0	0	0	1	1																
0	0	1	0	2																
0	0	1	1	3																
0	1	0	0	4																
0	1	0	1	5																
0	1	1	0	6																
0	1	1	1	7																
1	0	0	0	8																
1	0	0	1	9																
1	0	1	0	A																
1	0	1	1	B																
1	1	0	0	C																
1	1	0	1	D																
1	1	1	0	E																
1	1	1	1	F																

CL領域

GL領域

CR領域（未使用）

GR領域

左半分はASCIIと同じ

ISO/IEC 8859-1 （Latin-1）

収録されている文字（GR領域）

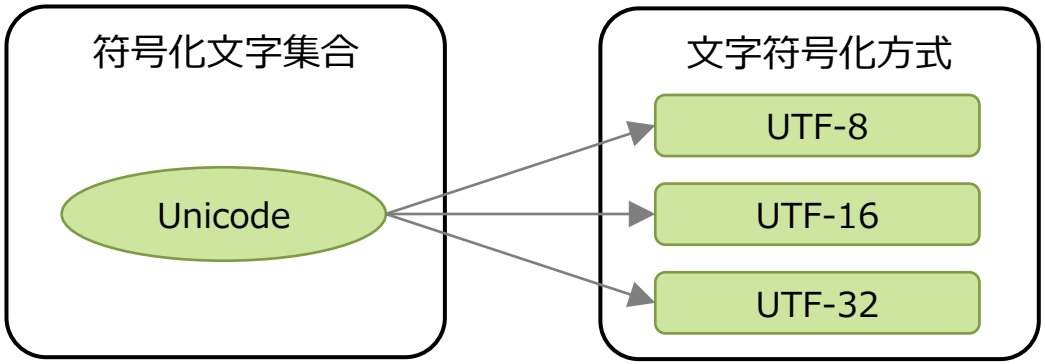
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Unicodeとその符号化方式

Unicode

世界で使われている全ての文字を共通の文字集合にて利用できるようにしようという考えで作られた符号化文字集合。
Unicodeは業界規格であり、ユニコードコンソーシアムが策定している。国際規格である**ISO/IEC10646**と互換がある。

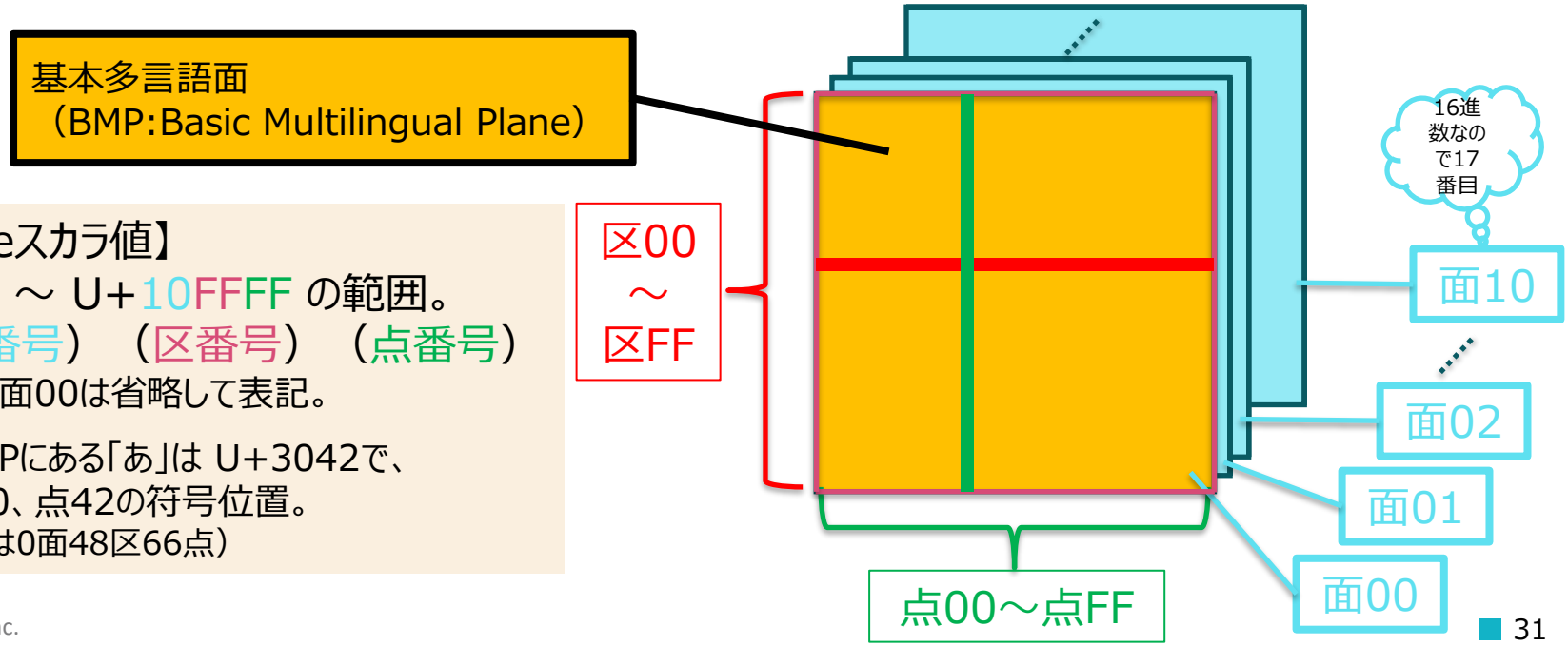
- 13版（2020年3月11日）では、世界各言語の143,859文字を収録。
- Unicodeの主な文字符号化方式には、UTF-8、UTF-16、UTF-32がある。



Unicodeの構造

Unicodeには、16ビットすなわち65,536の符号位置を持つ面が17面あり、合計100万あまりの符号位置（コードポイント）をもつ。符号位置は「U+4E00」のように接頭辞「U+」を付けた4～6桁の16進数で表記される。この表記をUnicodeスカラ値という。

最初の面00を基本多言語面（BMP:Basic Multilingual Plane）と呼び、日常的に用いる文字の大半がここに収められている。



【Unicodeスカラ値】
U+0000 ～ U+10FFFF の範囲。
U+（面番号）（区番号）（点番号）
※面番号で面00は省略して表記。

例えば、BMPにある「あ」は U+3042で、
面00、区30、点42の符号位置。
（10進数では0面48区66点）

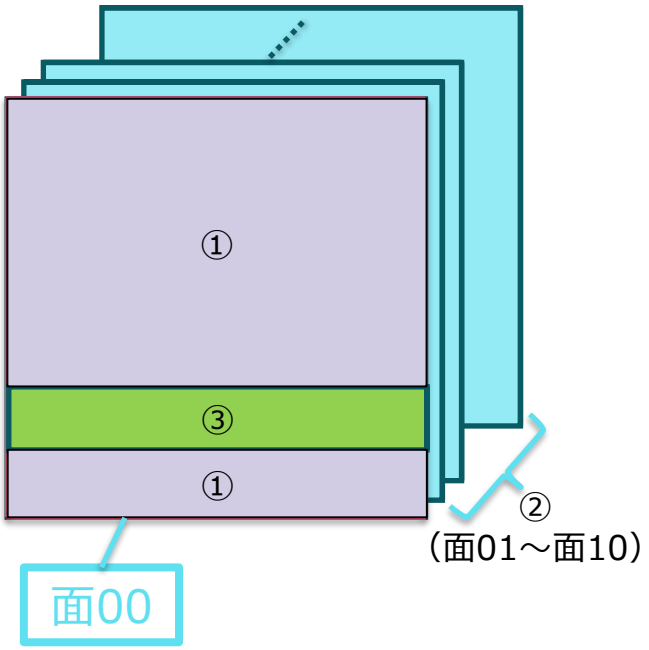
UTF-16

UTF-16は、16ビットを1単位として用いる文字符号化方式。BMPにある文字は、符号位置の整数を16ビットで表したビット組み合わせがそのままUTF-16の値になる。BMP以外にある文字は**サロゲートペア**と呼ばれる仕組みを用いて表現する。

① BMPの文字定義領域
（U+0000～U+D7FF、U+E000～U+FFFF）
16ビットで文字を表現。

② BMP以外の面（面01～面10）
32ビット（16ビット2つの**組み合わせ**）で表現。

③ サロゲートペア領域（後述）
BMPの中の文字が割り当てられていない領域。②に割り当てられた文字を表すためにこの符号位置の値を使う。



サロゲートペア

BMP以外の面の文字を表すために、上位サロゲートと下位サロゲートの組み合わせによって（面01～面10内の）1つの符号位置を示す仕組み。

- U+D800～U+DBFF：上位サロゲート
- U+DC00～U+DFFF：下位サロゲート

サロゲートは「代理」という意味。

UTF-16の計算方法

1. 符号位置が0xFFFF以下であればBMP内の文字なので、符号位置を16ビットとして表現。
2. 符号位置が0x10000以上であればサロゲートペアが必要。上位サロゲートと下位サロゲートを以下のように算出する。

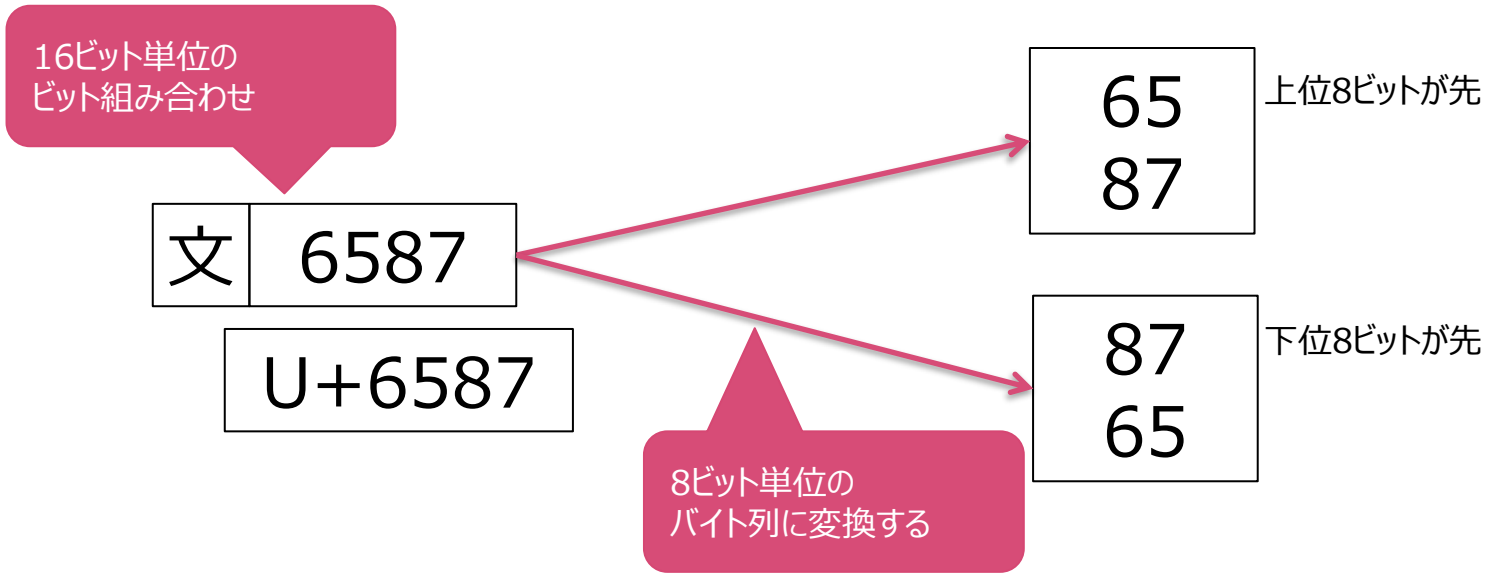
n = 符号化したい文字の符号位置
n′ = n − 0x10000 = yyyyyyyyyyxxxxxxxxxx　（x,yは2進数）
上位サロゲート = 110110yyyyyyyyyy
下位サロゲート = 110111xxxxxxxxxx

「鮭」＝U+29E3D の場合は、D867DE3Dとなる。

上位8ビット、下位8ビットどっちが先？問題

UTF-16は16ビット単位の符号のため、8ビット単位のバイト列にする場合、並び順の指定ができる。

- 上位8ビットから先に並べる方法
- 下位8ビットから先に並べる方法



Big Endian/Little Endian

UTF-16ではどちらを採用しても良い。

- 上位8ビットから先に並べることをBig Endianといい、Big EndianのUTF-16はUTF-16BEと呼ばれる。
- 下位8ビットから先に並べることをLittle Endianといい、Little EndianのUTF-16はUTF-16LEと呼ばれる。

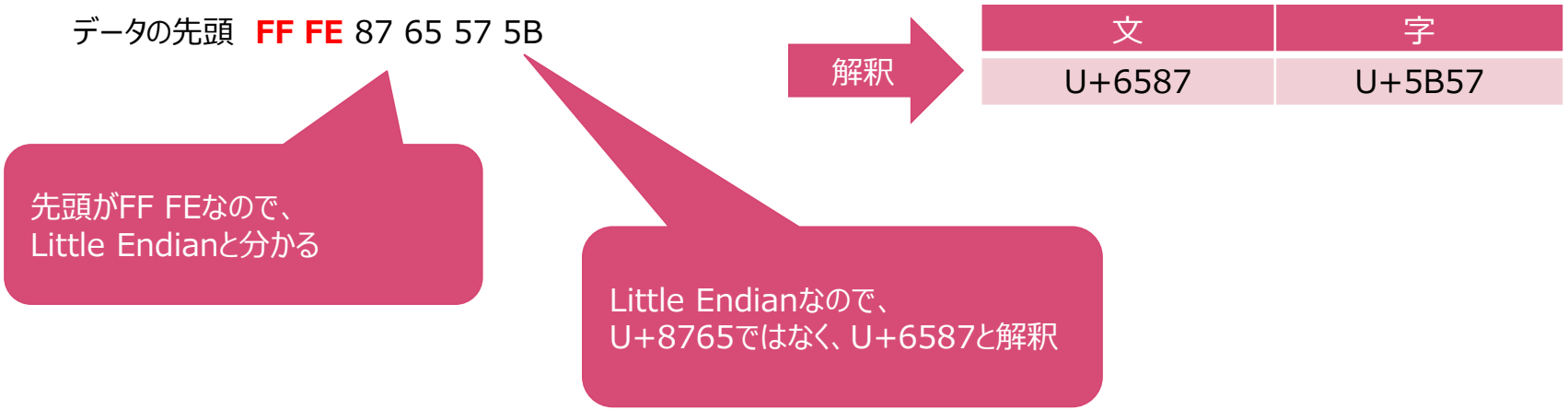
ちなみに、JIS X 0208のように第1・第2バイトが
あらかじめ決まっているものはこのような問題は発生しない。

BOM

バイト順マーク（Byte Order Mark：BOM）

どちらのバイト順を採用しているかを示す印（2バイト列）。

- データの先頭にFE FFというバイト列があればBig Endian
- データの先頭にFF FEというバイト列があればLittle Endian



UTF-8

ASCIIと互換性がある8ビット**単位**のUnicodeの文字符号化方式。

- 1つの符号位置の表現に1バイトから4バイトの可変長
- 元々ASCIIで構成されていたデータフォーマットやプロトコルを拡張することに向いている
- 漢字などは3バイト以上必要になるので従来のJIS系の文字符号化方式よりサイズが大きくなってしまう

ASCIIとの互換性

- ASCIIと同じバイト列を用いる
- 複数のバイトで1文字を表す際も2バイト目以降に0x00～0x7Fのバイトが出現することはない

→ 0x7F以下のバイトは常にASCIIとみなしてよい。

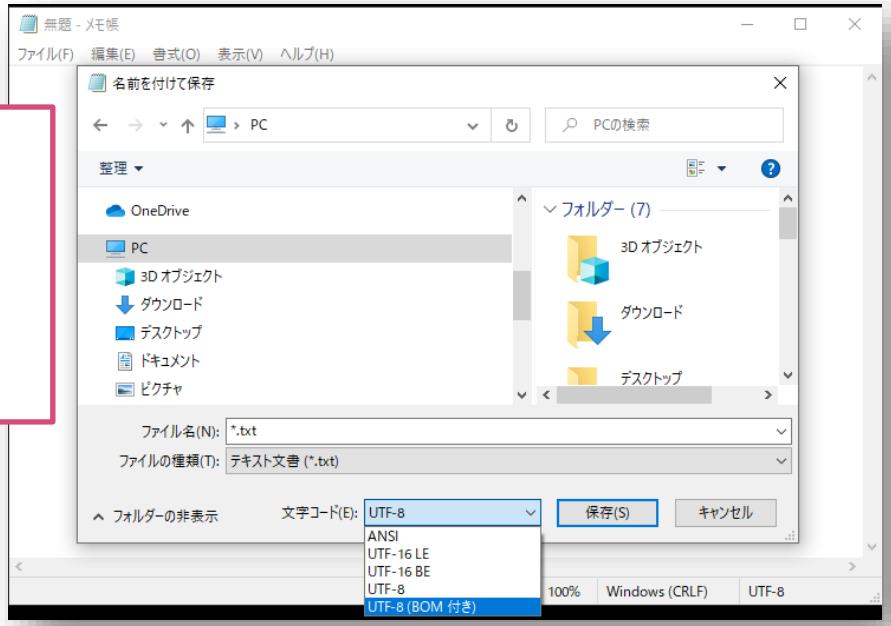
符号位置　16進数	UTF-8のバイト列　2進数
00000000～0000007F	**0xxxxxxx**
00000080～000007FF	110xxxxx 10xxxxxx
00000800～0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
00010000～0010FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

BOM付きUTF-8

本来は不要のBOMがデータの先頭についていることがある。
（本来はバイト順が問題にならない。）

UTF-8のBOMの値は：EF BB BF

UTF-8の先頭にBOMが付くことを想定していないプログラムの場合は予期せぬ結果になることがあるので注意が必要。



UTF-32

各符号位置が4バイト固定の文字符号化方式。
Unicode符号位置をそのまま32ビットで表現できる。

鮫：U+29E3D ⇒ 00 02 9E 3D

- BOM
  - Big Endian　：00 00 FE FF
  - Little Endian ：FF FE 00 00

ただし、00 になるバイトが多いため（75%を占める）、
記憶容量の無駄が多い。

ここまで学んだこと ～符号化文字集合と文字符号化方式～

TIS INTEC Group

符号化文字集合	文字符号化方式	説明
ASCII		米国のANSI規格。最も基本的な文字コード。
JIS X 0201		日本規格。ラテン文字と片仮名のふたつの文字集合を含む。
ISO/IEC 8859-1		ASCIIに加えて、アクセント記号付きアルファベット等を収録した西ヨーロッパ向けの規格。通称、Latin-1。
JIS X 0208		日本の漢字、平仮名、片仮名等を収録。第1・第2水準漢字を含む6,879文字。
	ISO-2022-JP	ASCII、JIS X 0201ラテン文字、JIS X 0208文字集合を扱う。電子メールで長年使用された符号化方式。
	EUC-JP	ASCIIとJIS X 0208文字集合を扱う。Unix上で利用されてきた符号化方式。
	Shift_JIS	JIS X 0201とJIS X 0208文字集合を扱う。Windows系システムや携帯電話で使用されてきた符号化方式。
JIS X 0213		JIS X 0208に足りない文字を追加した拡張版。 2000年初版。第3・第4水準漢字を含む11,233文字。
Unicode		世界中の文字を単一の符号化文字集合でカバーすることを目的とした規格。世界各言語の143,859文字を収録。
	UTF-8	Unicodeで一番利用される形式。1～4バイトの可変長。
	UTF-16	Unicodeを16ビットで表現。ただし、サロゲートペアは32ビット。
	UTF-32	Unicodeを32ビットで表現。

付録参照

プロジェクトにおける文字関連のタスク

プロジェクトにおける文字関連のタスク

プロジェクトのそれぞれのフェーズで、どのような作業を行っていけばよいのかを確認していきましょう。

- 要件定義
  - 非機能要件定義
  - 方式設計
- 設計
- PG/UT
- 結合テスト～システムテスト

非機能要件定義

主な実施事項

- 自システムで利用可能な文字集合を決定する。
- データ授受が発生する対外システムを洗い出す。
- 対外システムの利用可能な文字集合と文字コードを調査する。

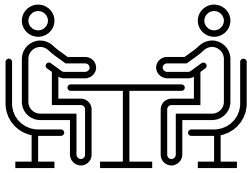
自システムで利用可能な文字集合を決定する

新システムの開発であっても、全くの新規ではなく、すでに現行の業務やシステムが存在していることが多い（つまり再構築）。
新システムでどのような文字集合を取り扱う必要があるのかを調査する。

分類	文字集合	ポイント
半角	記号	
	数字	
	英大文字	
	英小文字	ホストのシステムだと使用不可の場合がある。
	半角カタカナ	ホストのシステムだと使用不可の場合がある。
全角	非漢字	
	JIS第1水準漢字	
	JIS第2水準漢字	
	JIS第3水準漢字	「第3水準、第4水準」と「IBM拡張文字」は相性が悪い。 Shift-JIS系の文字コードでは、IBM拡張文字と 第3水準、第4水準のコード値が重なっているため。
	JIS第4水準漢字	
	IBM拡張文字	
	NEC特殊文字	JIS X 0213非漢字にも含まれている。
	外字	極力使わない。要件がある場合は個別に定義。

データ授受が発生する対外システムを洗い出す
対外システムの利用可能な文字集合と文字コードを調査する

対外接続先とやり取りする電文やファイルの
・文字集合
・文字コード（文字符号化方式）
をヒアリングする。



文字集合は、自システムとの差異を明らかにするために使うため、詳細にヒアリングする。

また、文字コードについても、後々の文字コード変換に必要となるため詳細にヒアリングする。
（例えば、「EBCDIC」と言っても、EBCDICカナ文字なのかEBCDIC英小文字なのか？まで確認する。）

方式設計

主な実施事項

- 自システムの文字コードを決定する。
- 対外接続における、互いの利用不可文字の扱いを決定する。
- 文字コード変換箇所、変換方法を決定する。

自システムの文字コードを決定

現在の主流はUnicode。
選択できるのであればUTF-8やUTF-16を選ぶのが定石。

UTF-8、UTF-16
を使っておけば、
たいてい何とかなる！

対外接続における、互いの利用不可文字の扱いを決定

接続先

うちのシステムでは
IBM拡張文字使ってます！
「髙」とかね。

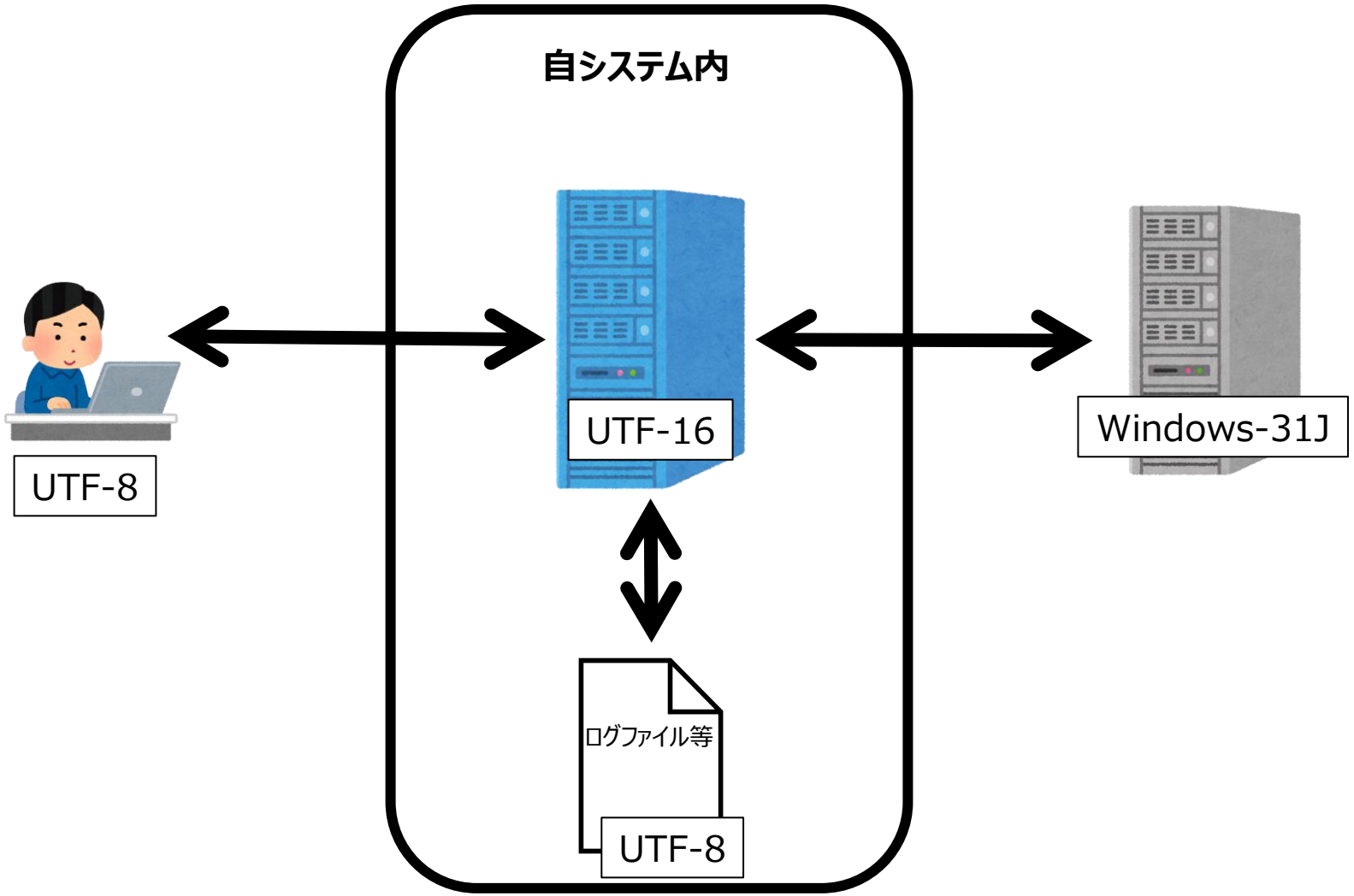
えっ？うちでは使えない。
どうしよう・・・。

自システム

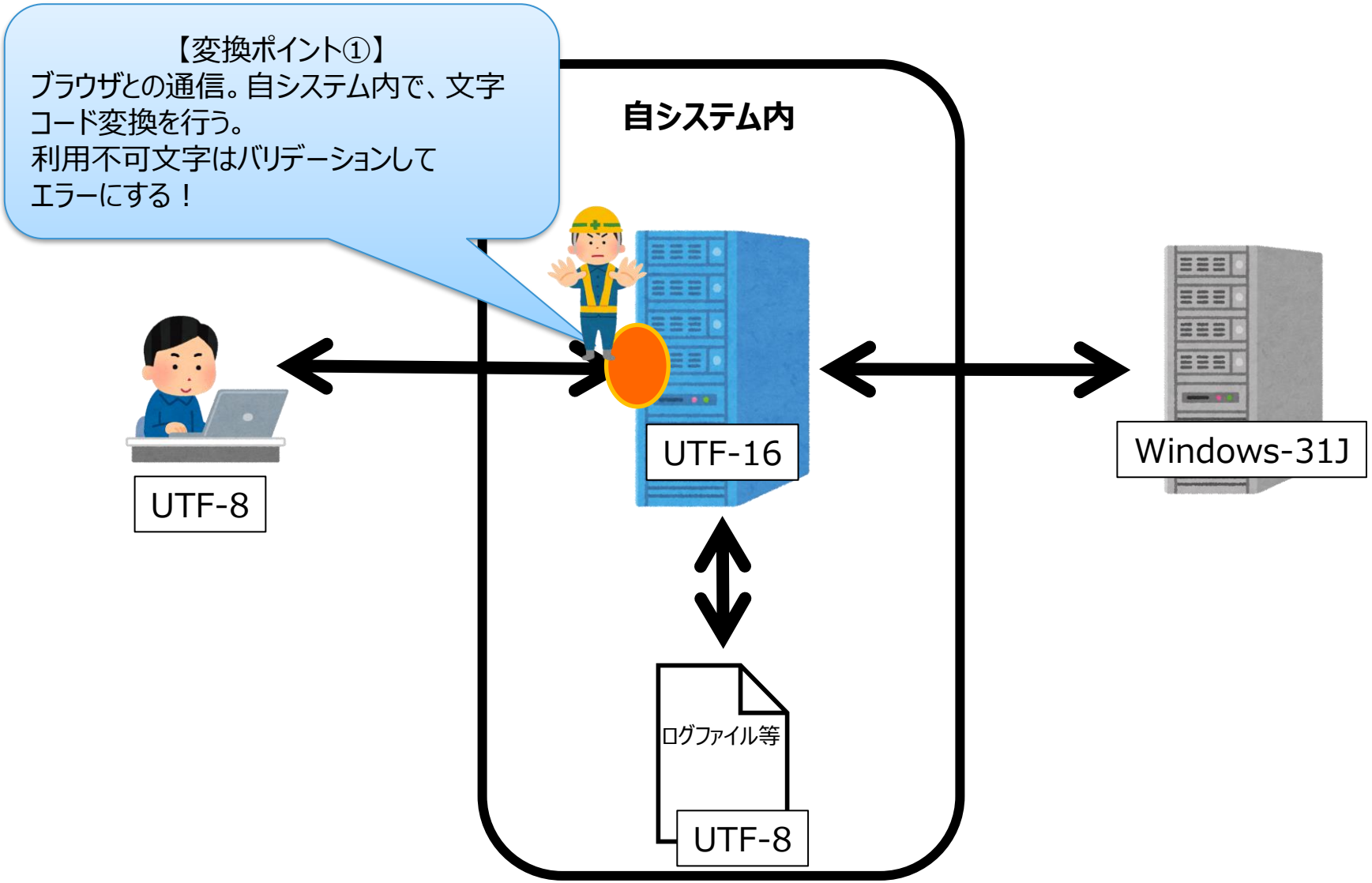
そんな時には。

使えない文字は、受取時にエラーとするか何かに置き換えるしかない。後工程で発覚すると信頼を損なうので、文字集合とあわせて、あらかじめ取り扱いを合意しておくことが大切。
例：使えない文字は、「〓」に変換。
「■」、「□」、「〓」、「●」がよく使われる。

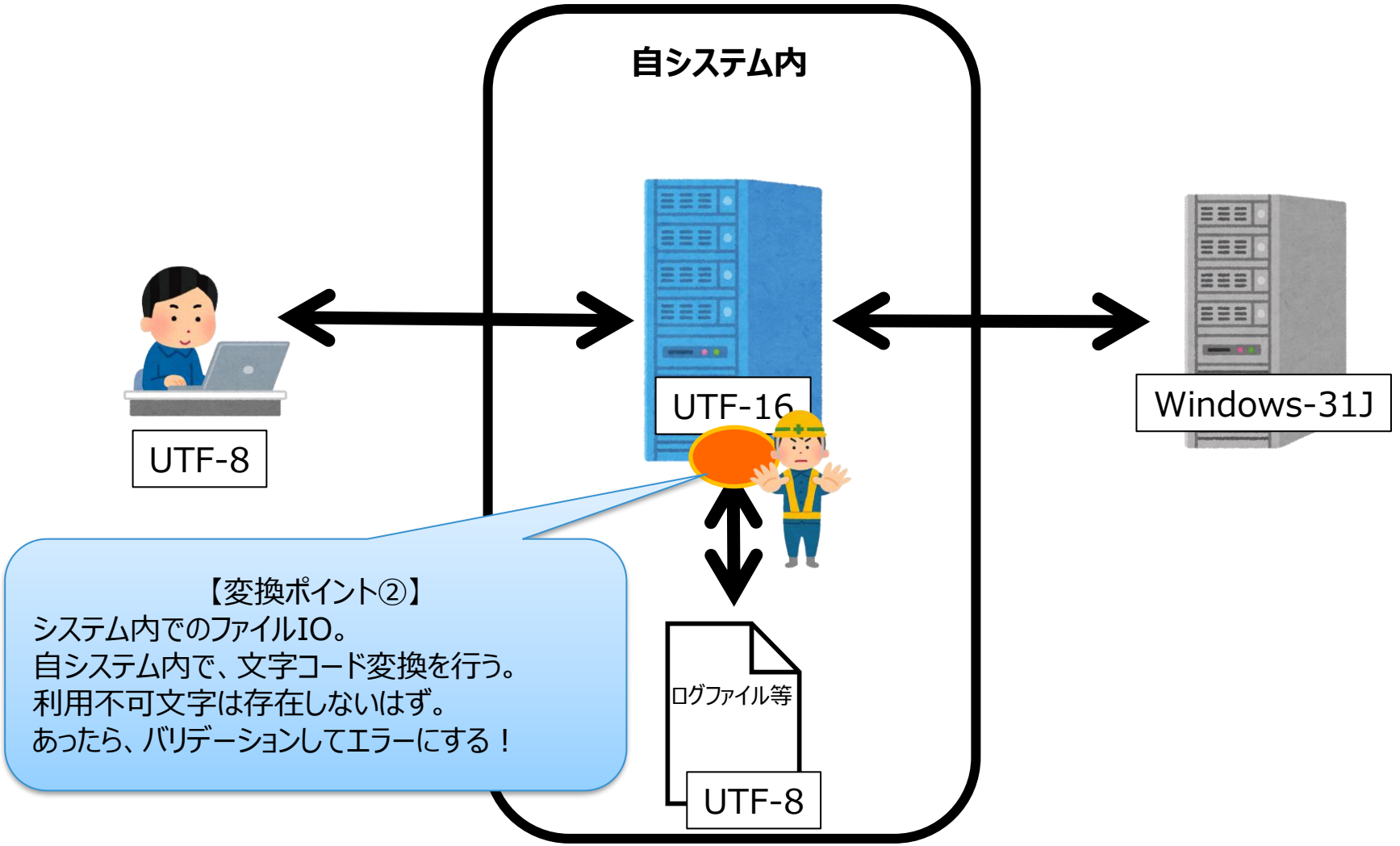
文字コード変換箇所、変換方法を決定



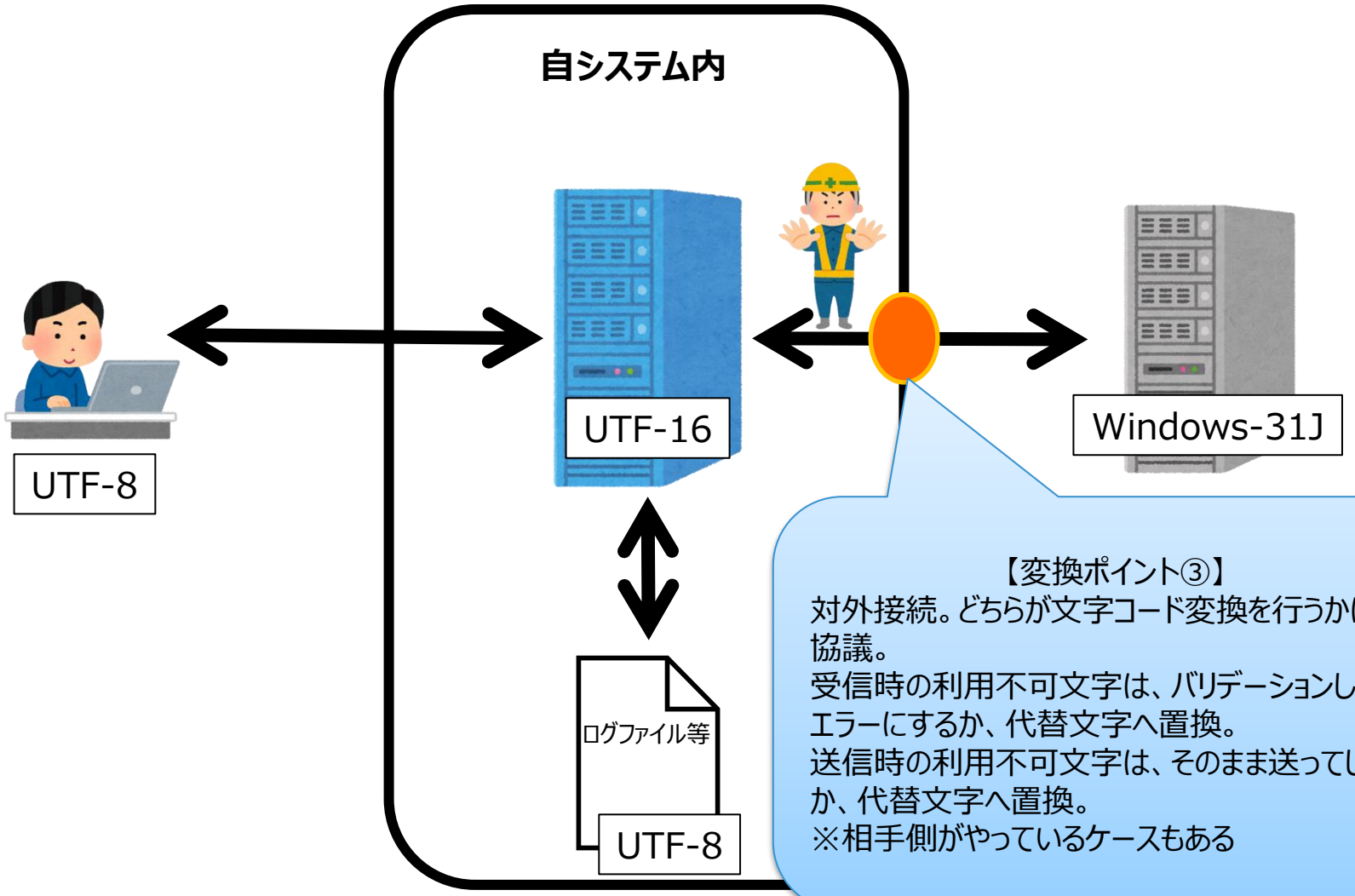
文字コード変換箇所、変換方法を決定



文字コード変換箇所、変換方法を決定

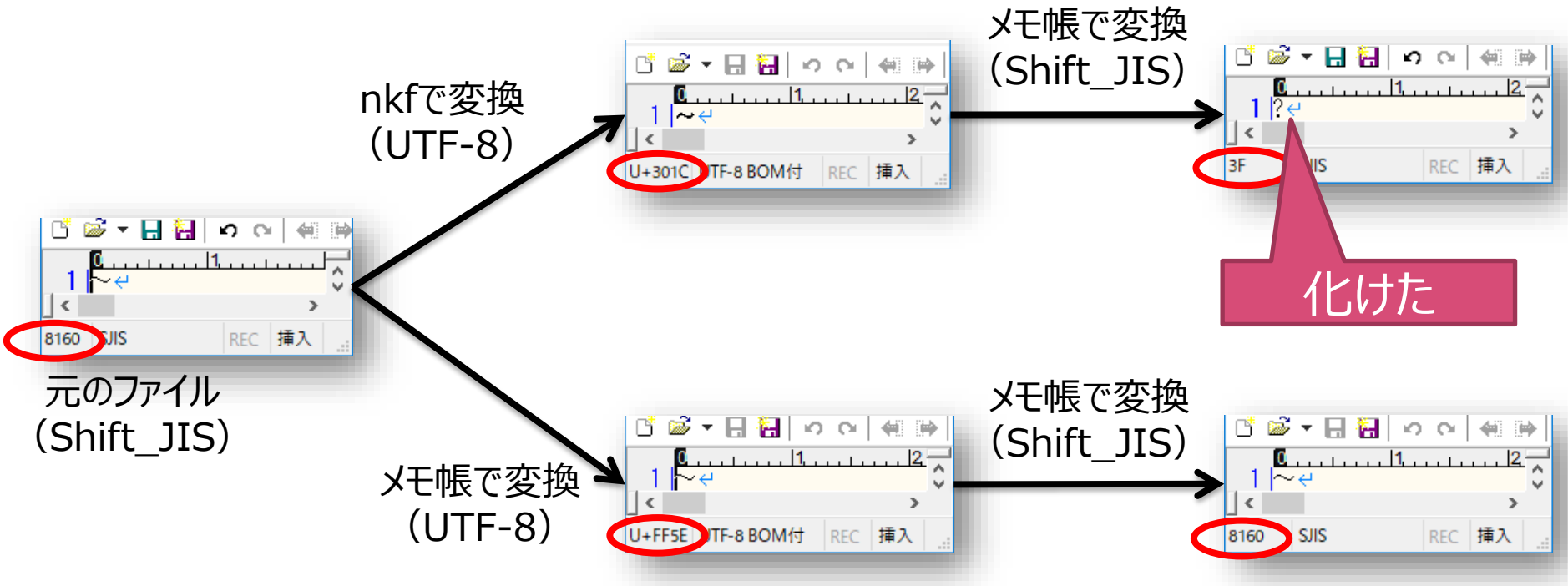


文字コード変換箇所、変換方法を決定



よくある変換時の問題

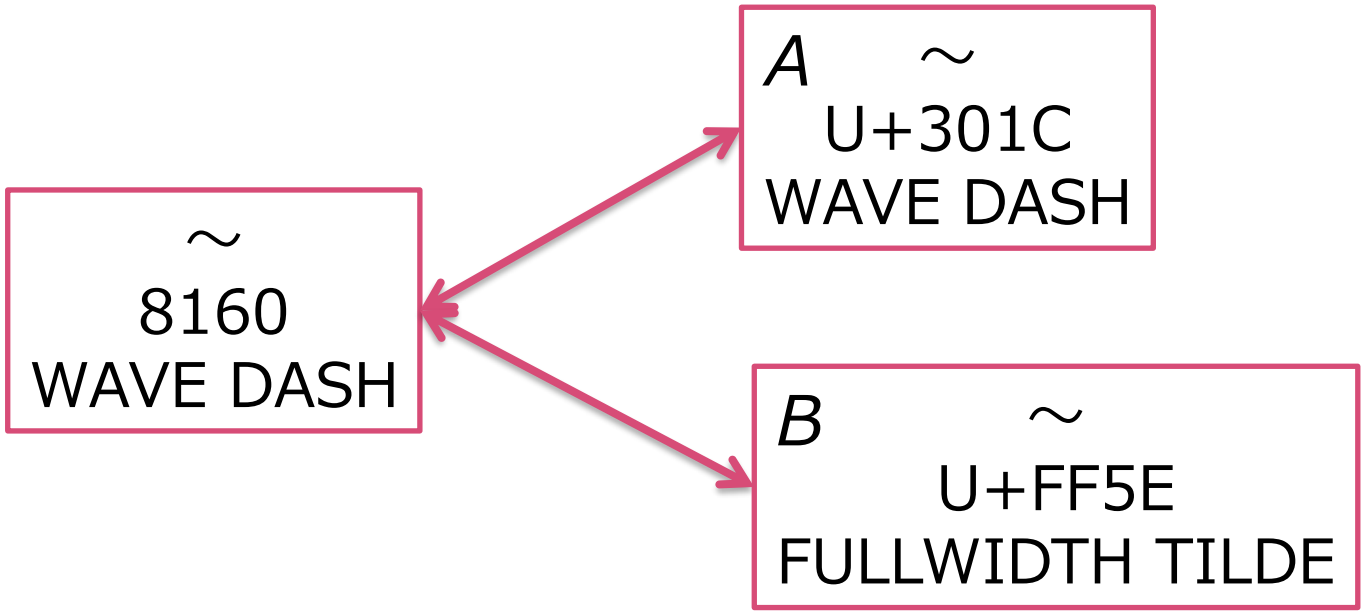
波ダッシュ（～）問題
JIS X 0208の1区33点にある記号「～」を、Unicodeに変換するとき、対応付けに2通りの実装があり、この為に生じる文字化け。



nkf:Network Kanji Filter
メモ帳：*Windows*に標準搭載されているメモ帳アプリ
文字コードの表示にはサクラエディタ（*https://sakura-editor.github.io*）を使用しています

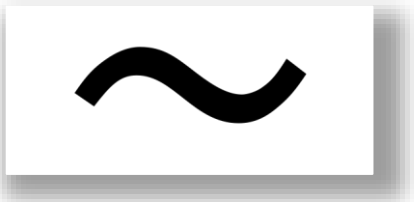
波ダッシュ（～）問題の原因

JIS X 0208の1区33点にある記号「～」を、Unicodeに変換するとき、対応付けに2通りの実装がある。

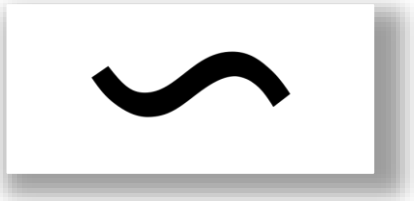


なぜ、U+FF5Eに対応付ける実装があるのか

Unicode のU+301Cの例示字形が（昔は）JISと違っていた。
（Unicode8.0（2014年8月）で訂正された）
一方、UnicodeのU+FF5Eは、字形的にはJISのWAVE DASHに近かった。
※所説あり



JISのWAVE DASH



（当初の）
UnicodeのWAVE DASH

https://commons.wikimedia.org/wiki/File:Wave_Dash.svg#mediaviewer/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Wave_Dash.svg

https://commons.wikimedia.org/wiki/File:Wave_Dash2.svg#mediaviewer/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Wave_Dash2.svg

波ダッシュ（～）問題の対処法

- コード変換をそろえる。
  Unicode⇔Shift_JISのコード変換仕様が異なるから化ける。これをそろえればよい。
- Unicode内で変換する。
  WAVE DASHに対応するUnicodeの符号位置が2つある事が問題。どちらかに寄せてしまい、他方が現れたら、寄せたほうにUnicode内で変換する。

状況に合わせて、適切な方法を選択する。

文字コード変換方法の決定の実例

ファイル入出力をするに当たり SJIS や MS932 から Unicode への変換が2通り存在する文字（以下の7文字）について、システムで使用可能な文字を一方に統一する。

○ 取り扱う文字

FULLWIDTH TILDE	～	U+FF5E
PARALLEL TO	∥	U+2225
FULLWIDTH HYPHEN-MINUS	－	U+FF0D
FULLWIDTH CENT SIGN	￠	U+FFE0
FULLWIDTH POUND SIGN	￡	U+FFE1
FULLWIDTH NOT SIGN	￢	U+FFE2
HORIZONTAL BAR	―	U+2015

× 扱わない文字

WAVE DASG	～	U+301C
DOUBLE VERTICAL LINE	‖	U+2016
MINUS SIGN	−	U+2212
CENT SIGN	¢	U+00A2
POUND SIGN	£	U+00A3
NOT SIGN	¬	U+00AC
EM DASH	—	U+2014

– SJIS、MS932 などのUTF-8で扱う文字種よりも狭い文字コードのファイルを出力する場合、出力文字コードにて存在しない文字は "■"に変換して出力を行う。

設計、PG/UT

主な実施事項

- 文字のバリデーション、文字コード変換処理を設計する。
- 文字のバリデーション、文字コード変換処理を実装・テストする。

設計、PG/UT

通常、文字コード変換は各業務アプリで個々に設計や実装をしない。

方式設計で決めた方法に従って
設計・実装をすること！

方式でも、あまり自前でやらずに、言語仕様やフレームワークの力を借りる方法をとることが多い。
もし自前で変換ロジックを組む場合、性能に注意する。

結合テスト以降

きちんと設計出来ていれば、
文字化けなんか起こるわけがない。

それでも起きてしまった場合は、

（1） IN/OUTの文字をコード値（16進数）で確認
（2） INの文字（16進数）は想定内か？OUTの文字（16進数）は想定か？
（3） INが悪ければ、INの情報の作成元を調査
（4） OUTが悪ければ、文字コード変換処理を調査

16進数で確認することが
重要！

まとめ

プロジェクトにおける文字関連のタスク

工程	分類	タスク
要件定義	非機能要件定義	自システムで利用可能な文字集合を決定
		データ授受が発生する対外システムの洗い出し
		対外システムの利用可能な文字集合と文字コードを調査
	方式設計	自システムの文字コードを決定
		対外接続における、互いの利用不可文字の扱いを決定
		文字コード変換箇所、変換方法を決定
設計	設計	文字バリデーション、文字コード変換処理を設計
PG/UT	PG/UT	文字バリデーション、文字コード変換処理を実装・テスト
結合テスト	対外接続テスト	文字化け等の確認

ITで、社会の願い叶えよう。

TIS INTEC
Group

付録　SHIFT_JIS

Shift_JIS

JIS X 0201の8ビット符号の隙間（未定義位置）に、
JIS X 0208を一定の計算式で変換したものを入れ込んだ
文字符号化方式。

				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1	0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	制御文字		JIS X 0201 ラテン文字集合						隙間		JIS X 0201 片仮名集合				隙間	
0	0	0	1	1																
0	0	1	0	2																
0	0	1	1	3																
0	1	0	0	4																
0	1	0	1	5																
0	1	1	0	6																
0	1	1	1	7																
1	0	0	0	8																
1	0	0	1	9																
1	0	1	0	A																
1	0	1	1	B																
1	1	0	0	C																
1	1	0	1	D																
1	1	1	0	E																
1	1	1	1	F																

Shift_JIS

2バイトコードの場合、第1バイト目に符号表で図形文字の割り当てのない部分（0x81～0x9Fと0xE0～0xEF）を使う。
第2バイト目は（0x40～0x7Eと0x80～0xFC）を使う。

<2バイトコードの1バイト目>

				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1	0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0										■					■	
0	0	0	1	1									■	■					■	
0	0	1	0	2									■	■					■	
0	0	1	1	3									■	■					■	
0	1	0	0	4									■	■					■	
0	1	0	1	5									■	■					■	
0	1	1	0	6									■	■					■	
0	1	1	1	7									■	■					■	
1	0	0	0	8									■	■					■	
1	0	0	1	9									■	■					■	
1	0	1	0	A									■	■					■	
1	0	1	1	B									■	■					■	
1	1	0	0	C									■	■					■	
1	1	0	1	D									■	■					■	
1	1	1	0	E									■	■					■	
1	1	1	1	F									■	■					■	

<2バイトコードの2バイト目>

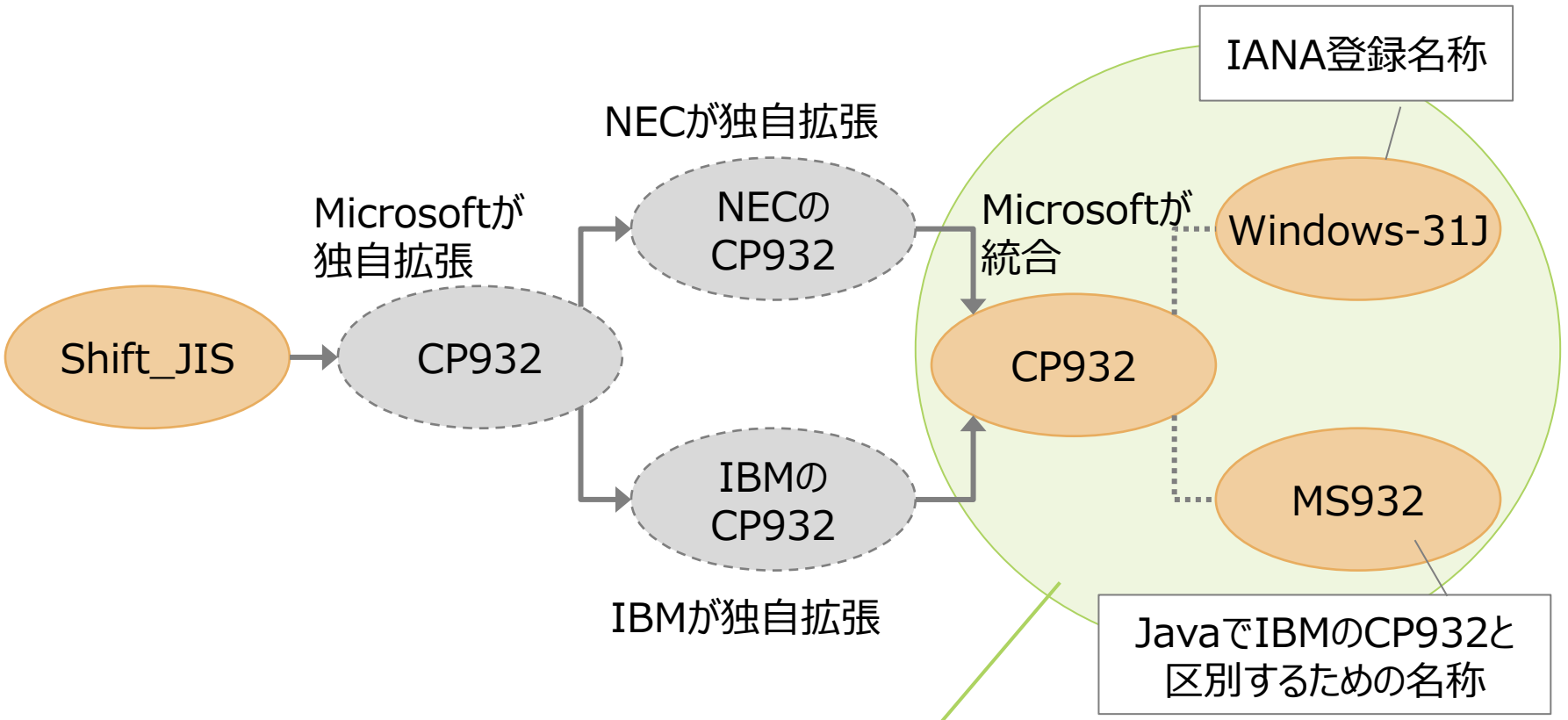
				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1	0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0					■	■	■	■	■	■	■	■	■	■	■	■
0	0	0	1	1					■	■	■	■	■	■	■	■	■	■	■	■
0	0	1	0	2					■	■	■	■	■	■	■	■	■	■	■	■
0	0	1	1	3					■	■	■	■	■	■	■	■	■	■	■	■
0	1	0	0	4					■	■	■	■	■	■	■	■	■	■	■	■
0	1	0	1	5					■	■	■	■	■	■	■	■	■	■	■	■
0	1	1	0	6					■	■	■	■	■	■	■	■	■	■	■	■
0	1	1	1	7					■	■	■	■	■	■	■	■	■	■	■	■
1	0	0	0	8					■	■	■	■	■	■	■	■	■	■	■	■
1	0	0	1	9					■	■	■	■	■	■	■	■	■	■	■	■
1	0	1	0	A					■	■	■	■	■	■	■	■	■	■	■	■
1	0	1	1	B					■	■	■	■	■	■	■	■	■	■	■	■
1	1	0	0	C					■	■	■	■	■	■	■	■	■	■	■	■
1	1	0	1	D					■	■	■	■	■	■	■	■	■	■	■	
1	1	1	0	E					■	■	■	■	■	■	■	■	■	■	■	
1	1	1	1	F					■	■	■		■	■	■	■	■	■	■	

区点番号からShift_JISのコードポジション変換

Shift_JISは第1バイトの範囲を狭めて、第2バイトを広げているため、EUC-JPと異なり区番号・点番号がそれぞれ第1・第２バイトに直接相当しない。
→計算式を用いて変換している。

区番号：k、 点番号：t、 ※（k-1） ÷ 2は小数点以下切り捨て
第1バイト
 1≦k≦62のとき （k-1） ÷ 2 + 0x81
63≦k≦94のとき （k-1） ÷ 2 + 0xC1
第2バイト
・kが奇数の場合
 1≦t≦63のとき t + 0x3F
64≦t≦94のとき t + 0x40
・kが偶数の場合
 t +0x9E

Shift_JISを拡張した文字コード



現在の「CP932」「MS932」「Windows-31J」は同じ文字コードを指す。
扱いとして以下のように考えるとよい。
Shift_JIS ≒ CP932（Microsoft統合） ＝ MS932 ＝ Windows-31J

付録　EUC-JP

EUC-JP

- Extended Unix Code
  - Unix系OSで使用されている
- ASCIIとJIS X 0208を同時に用いる8ビットの文字符号化方式
  - 双方に存在するラテン文字や数字は2つのコードを持つことになり、一意な符号化という点で原則に反してしまう。
    →EUC-JPでは、ASCIIの方を用いるようにしている。

EUC-JP

GL領域はASCIIで固定、GR領域は制御文字で切り替える。

				b8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b4	b3	b2	b1	0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0																
0	0	0	1	1																
0	0	1	0	2																
0	0	1	1	3																
0	1	0	0	4																
0	1	0	1	5																
0	1	1	0	6																
0	1	1	1	7																
1	0	0	0	8																
1	0	0	1	9																
1	0	1	0	A																
1	0	1	1	B																
1	1	0	0	C																
1	1	0	1	D																
1	1	1	0	E																
1	1	1	1	F																

ASCIIで固定

JIS X 0208
JIS X 0201
（片仮名集合）
JIS X 0212
（補助漢字）
を切り替える

EUC-JP

- 切替方法
  - GR領域にはJIS X 0208が呼び出されている
  - 制御文字SS2（0x8E）の**直後の1文字**はJIS X 0201
  - 制御文字SS3（0x8F）の**直後の1文字**はJIS X 0212
  - 1文字過ぎた後は自動的にJIS X 0208に戻る

CC A8	A4 CE	43	61	66	8F	AB B1	A4 CD	A1 A3
岬	の	C	a	f	SS3	é	ね	。

0xA0以上（GR領域）の値は
JIS X 0208を表す

制御文字『SS3』直後
1文字分はJIS X 0212を表す

自動的にJIS X 0208
に戻る

付録　ISO-2022-JP

ISO-2022-JP

- ASCII、JIS X 0201ラテン文字、JIS X 0208を切り替えて使用する7ビットの文字符号化方式。
- 8ビット符号表の右半分を使用せず、エスケープシーケンスを用いてGL領域の文字集合を切り替える。

ISO-2022-JP

符号化文字集合	エスケープシーケンス	文字列表現
ASCII	1B 28 42	ESC (B
JIS X 0201 ラテン文字	1B 28 4A	ESC (J
JIS X 0208 1978年版	1B 24 40	ESC $ @
JIS X 0208 1983年版	1B 24 42	ESC $ B

改行の前にはASCIIまたは
ラテン文字に戻すルール

1B 24 42	3A 23	46 7C	24 4F
83JIS の指示	今	日	は

終端ではASCIIに
戻すルール

次のエスケープシーケンスが出るまで同じ文字集合を使う

1B 28 42	33	31	1B 24 42	46 7C	1B 28 42
ASCII の指示	3	1	83JIS の指示	日	ASCII の指示