地質調査所ネットワークで推奨する情報交換用漢字コードとコード変換プログラムについて

野呂春文（地質情報センター）・川勝　均（地殻熱部）・宮崎光旗（地殻物理部）
Harufumi NORO　Hitoshi KAWAKATSU　Teruki MIYAZAKI

1. はじめに

工業技術院筑波センター内に　イーサネットがはりめぐらされ　地質調査所の所内のネットワークも　ほぼ完成しました.　各地の研究機関や大学との情報流通路である JUNET の利用も　可能になりました.

今日では　計算機のネットワークは　世界中に広がっています.　商業的でないものに限ってみても　全世界に広がった BITNET　日本国内の JUNET　北米に広がった CSnet・ARPA インターネット・UUCPnet・UUnet ヨーロッパの　EUnet　等があります.　これらのネットワークは　相互に結合されており　地域的な情報交換に限られず　国際的な情報交換もできるようになっています.　たとえば　日本国内の JUNET は　東京大学をゲートウェイにして CSnet につながり　CSnet は北米大陸の　いくつもの　ネットワークに　つながっています.

おもに計算機関係の　研究者・技術者の間では電子メイルを用いた　個人間の情報交換や　ニュース・掲示板システムによる討論や会議が　常識になりつつあります.　もう少し小さな話ですが　一つの組織の中で　日常配布される大量の文書を　ネットワークに乗せることによって　紙の洪水を防ごうとしているところもあります.

このような状況ですので　地質調査所の内外に所内ネットワークや　JUNET を用いて　ニュース・掲示板・電子メイルのやりとり等を　おこないたいと考えている人がたくさんいると思います.　いわゆるワークステーションなら　当然これらをこなすためのプログラムが用意されていますし　最近ではパソコン用でも質の高い通信ソフトウェアがたくさん流通しているので　簡単そうに思えます.

ところが　大問題があるのです.　計算機の漢字コードの不統一です.　ニュースでも　掲示板でも　メイルでも　送る　受け取る　そして　眺める　のは簡単ですが　そのままでは読めないことが多いのです.

送り手と受け手の漢字に関する取り決めが一致しなければ　読むことができないのですから当然です.　もちろんそれらの文書を全部英語にするとか　全部ローマ字にするとか　とにかく漢字とひらがなを使わないようにすれば問題は起こりませんが　日本にいるのに読みやすい漢字かな混じり文が使えないのは　なんのためのネットワークかわかりません.

このような問題は　何もネットワークに限るわけではありません.　計算機に限るわけでもありません.　たとえ　ミツバチの世界でも情報の交換が成立するためには送り手と受け手の間で　情報について取り決めが一致していなければならないのは　当然ですから.　とにかく　最低限　情報を構成する符号について一致していなければ情報交換は不可能です.　文字　音素　身振り……

というわけで　所内　所外の情報流通を容易にするという立場からの最低限の取り決めとなる　地質調査所ネットワークで推奨する情報交換用漢字コードについて説明します.

2. 地質調査所ネットワークの推奨漢字コード

地質調査所ネットワークで　電子メイル　ニュース　掲示板等の機能を利用するために用いる情報交換用漢字コードとして JIS 7 ビット漢字コードを推奨します.

ここで JIS 7 ビット漢字コードとは JIS C6220 および JIS C6226 に定められた 7 単位コードです.　そして　漢字を使うための取り決めは JIS C6228 にしたがうものとします.　すなわち「今から漢字だよ」という指示は ESC $ @ 16進で　1b　24　40　または ESC $ B 16進で　1b　24　42とします.　ここで ESC はエスケープコード　すなわち16進の 1B　10進の27です.　二番目の方が新しい JIS 規格ですが　旧来の文書との互換性を保つため　どちらでも良いことにします.　そして「今から英数字だよ」という指示は ESC（J　16進で　1b　28　4a とします.　パソコン通信等で　かつて用いられた ESC（H は誤りですので　使うべきでありません.

ここで言う英数字は　JIS 英数字ですので　いわゆる ASCII コードとほとんど同じものです.　主として　大型計算機で用いられている EBCDIC コードは事実上の標準コードのひとつではありますが　もともと IBM 社

	0	1	2	3	4	5	6	7
0		D_E		0	@	P		p
1	S_H	D_1	!	1	A	Q	a	q
2	S_X	D_2	"	2	B	R	b	r
3	E_X	D_3	#	3	C	S	c	s
4	E_T	D_4	$	4	D	T	d	t
5	E_Q	N_K	%	5	E	U	e	u
6	A_K	S_N	&	6	F	V	f	v
7	B_L	E_B	'	7	G	W	g	w
8	B_S	C_N	(	8	H	X	h	x
9	H_T	E_M	)	9	I	Y	i	y
A	L_F	S_B	*	:	J	Z	j	z
B	H_M	E_C	+	;	K	[	k	{
C	C_L		,	<	L	¥	l	¦
D	C_R		−	=	M	]	m	}
E	S_O		.	>	N	^	n	~
F	S_I		/	?	O	_	o	DEL

図1　JIS 7単位符号表です．16進で00から1Fまでは 特別の意味を持つ 各種の制御コードで 20は空白（スペース）21から7Eまでが 普通の文字 そして 7Fは一文字削除のコードです． ASCII コードとの違いは わずかで ASCII の＼が JIS の¥におきかわっているだけです． 英数字と漢字を混在させるためには エスケープシークエンスによる切り替えが必要です．

の社内コードであり しかも 機種による変化があるため JIS 規格にはなっていません． 以上を推奨する理由 そしてその細部は 次に述べます．

3. JIS 7ビット 漢字コードとは

上で述べたように 情報交換のための符号として JIS C6220（情報交換符号）の7単位符号と8単位符号 JIS C6226（情報交換用漢字符号）があります． これが コードの規格です． そしてそのコードを使うための規格が JIS C6228です． なお JIS ではビットを単位 コードを符号 と呼んでいます． 以下の文章では 両方の用語を区別せずに使います．

これらの規格は 英数字と漢字その他の混在した文章を計算機の間で交換するために定められたもので 単に日本の国内だけの取り決めではなく 世界中で通用するものです（ISO 規格です）． 英数字と漢字とアラビア文字の混在した文章 日本語漢字と中国語漢字とハングルの混在した文章 というような文章を計算機の間で交換できるように考えられています．

なぜ 文字の混在が問題かというと 英数字だけなら1バイトだけで1文字が表現できるのに対して 漢字などは字種が多いので1文字を表現するのに複数バイトを必要とするからです． つまり 1文字を構成するバイト数が異なるような文字が混在し しかも場合によって同じコードが別の文字を意味することもある というような文書を取り扱わねばならないわけです．

よほどキチンとした取り決めがないと まったくバベルの塔の現代版です． そういうわけで ISOの国際的な取り決めが作られたわけです．

日本語漢字を使う場合の規格の例を示します． たとえば JIS C6226に定められた漢字コードを 16進の20から7F（通常 英数字のある場所です）に割り当てて使う場合 ESC $ @であらわされるエスケープシークエンス（エスケープ文字列 とも言います）を送ります． すると そのエスケープシークエンス以降に現れるコードは2バイトずつの単位で JIS C6226での漢字であると解釈されます． なお新しい JIS では ESC $ Bですが普通はどちらも使われています． 漢字の使用を終わって JIS 英数字を使う場合には ESC（Jであらわされるエスケープシークエンスを送ります． これ以降にあらわれるコードは 1バイトずつ英数字と解釈されます．なお このエスケープシークエンスを「漢字の使用を終わる」 と解釈して「漢字アウト」と呼ぶことがありますが それは誤りです． このエスケープシークエンスの意味は「今から JIS 英数字を使う」ということです．たとえばハングルを使っていて JIS 英数字に移る場合を考えてみれば この違いがおわかりになると思います．

以上が JIS 規格の概要の“あらすじ”ですがここまで7ビットと8ビット あるいは7単位と8単位というものについて なにも説明しませんでしたのでここで簡単にふれます．

7単位符号とは 16進の20から7Fまで すなわち7ビットだけで表現できるコードを 文字として使うということです（図1を見て下さい）． したがって 英数字と漢字を使うには 上で述べたようなエスケープシークエンスによる切り替えが必要です．

一方 8単位符号とは16進で20から7Fまでと A0からFFまでの8ビットのコードを文字として使うということです． 通常 日本では A0 から FF までの領域には 半角のカタカナを置いています（図2を見て下さい）． 今日では 半角のカタカナはほとんど使いませんから この領域に漢字を割り当てれば 英数字と漢字がエスケープシークエンスによる切り替え無しで混在できます． この方法を実現するには ESC $）@であらわされるエスケープシークエンスを送ります． そして漢字をあらわすコードの頭のビットをすべて1にするすなわち JIS C6226コード表のコードのすべての1バイトずつに 16進の80を加えておきます． すると 先

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p				ー	タ	ミ		
1	S_H	D_1	!	1	A	Q	a	q			。	ア	チ	ム		
2	S_X	D_2	"	2	B	R	b	r			「	イ	ツ	メ		
3	E_X	D_3	#	3	C	S	c	s			」	ウ	テ	モ		
4	E_T	D_4	$	4	D	T	d	t			、	エ	ト	ヤ		
5	E_Q	N_K	%	5	E	U	e	u			・	オ	ナ	ユ		
6	A_K	S_N	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	B_L	E_B	'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8	B_S	C_N	(	8	H	X	h	x			ィ	ク	ネ	リ		
9	H_T	E_M	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	L_F	S_B	*	:	J	Z	j	z			ェ	コ	ハ	レ		
B	H_M	E_C	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	C_L		,	<	L	¥	l	¦			ャ	シ	フ	ワ		
D	C_R		−	=	M	]	m	}			ュ	ス	ヘ	ン		
E	S_O		.	>	N	^	n	~			ョ	セ	ホ	゛		
F	S_I		/	?	O	_	o	DEL			ッ	ソ	マ	゜		

図２　JIS ８単位符号表です．　16進で00から１Ｆまで　そして　80から９Ｆまでが制御コードです．　空白も文字として勘定すると　20から７Ｅ　そして　A0 から FE までが文字で　７Ｆが一文字削除コード　FF　は使いません．　普通は　21から７Ｅまでに英数字をわりあて　Ａ１からDFまでに半角のカタカナをわりあてます．　エスケープシークエンスなしで　英数字と漢字を混在させるには　半角カタカナのある場所　つまり　Ａの列からＦの列までに漢字を割り当てます．　拡張ユニックス漢字コードで　採用しているのがその方法です．

頭ビットが０であれば英数字　先頭ビットが１であれば漢字と解釈されるようになります．　ユニックスワークステーションの多くで採用されている拡張ユニックス漢字コード（EUCコード）とよばれるものは　はじめからこの状態にあると考えることができます．

4.　なぜ　JIS　７ビット　漢字コードなのか

以上見てきたように　７ビットコード（７単位符号）にくらべて８ビットコード（８単位符号）は　ずっと便利です．　なのに　なぜ７ビットコードを提案するのか疑問に思われるでしょう．その疑問に　まず答えましょう．

第一の理由は　７ビットコードなら現在のいかなる計算機でも　文字として扱ってくれるからです．　８ビットコードで　16進で80以上はかなり多くのプログラムの中で　文字として認識されないことがあります．　計算機に付随する　あるいは計算機と同程度に重要な外部入出力機器も同様です．　７ビットコードなら　最悪の場合でも文字化けが起きるだけで　画面がメチャクチャになったり　プリンターがでたらめに動き出すというようなことが防げます．

第二の理由は（実は　こっちが本音なのですが）JUNET が　JIS ７ビット漢字コードを用いているからです．広い範囲の研究機関で　情報のやりとりに使われている JUNET で用いている漢字コードと地質調査所における情報交換に用いる漢字コードが同じほうが良いのは　当然です．

次のような場合を想定すれば　上の理由を理解していただけると思います．　日本から　アメリカ合衆国にいる日本人にメイルを送るとします．　メイルは　おそらく JUNET で東京大学に着き　そこから CSnet でアメリカ合衆国に至り米国内のネットワークで届先に着くという経路をたどるか　あるいは JUNET で KDD 内の Inetclub に着き　UUnet で合衆国に至り米国内ネットワークで届先に着く　という経路をたどるでしょう．　CSnet への加入者であれば　JUNET を経由せずに直接 CSnet に入るかもしれません．

とにかく　いずれの経路をたどるにしろ　もしも８ビットコードの文書を送ったとしたら　途中のネットワークを管理するプログラムは　８ビット目を保存しないと考えたほうが良いでしょう．　最悪の場合８ビット目はぎ取られてしまいます．　文書が　すべて　英数字でできているのなら８ビット目が　はぎとられても何も問題がありません．　しかし　相手が日本人ですから　多分漢字かな混じり文のメイルでしょう．　８ビットコードの漢字を含んでいるので　８ビット目をはぎ取られると　もとの文書は復元されません．　広域　あるいはグローバルなネットワークで情報を交換するには　当面　７ビットコードを用いるしかないのです．

つぎのような疑問を持たれる方もいるでしょう．「シフト JIS 漢字というものがある．　これならパソコンで使っているコードだから　何も面倒なことがないし　英数字と漢字が切り替えなしで共存できるのだからずっと良いではないか」

この疑問に対しては　シフト JIS 漢字とよばれるものは　８ビットコードであり　上で述べた第一の理由に反するということ　またこの符号系は　マイクロソフト社の社内規格であって　国際的に認知されたものでないためグローバルな情報交換を阻害するおそれがあること　英数字と漢字だけならいいが　例えばハングルが入れば対処できないこと　等の問題があるということを指摘して答えとします．　なお　従来パソコン通信では　シフト JIS 漢字が広く使われてきましたが　徐々に JIS 漢字コードに移行しつつあります．

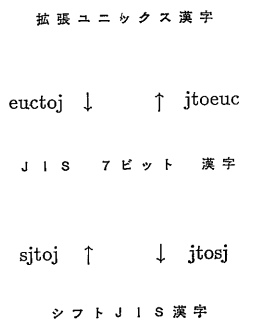

図3　JIS　7ビット　漢字コード・拡張ユニックス漢字コード・シフト JIS 漢字コードの文書を　相互に変換するためのフィルタープログラムの役割の説明です.　JIS　7ビット漢字コードと拡張ユニックス漢字コードを変換するのが euctoj と jtoeuc のふたつのフィルターです.　そして JIS 7ビット漢字コードとシフト JIS 漢字コードの文書を変換するのが sjtoj と jtosj のふたつのフィルターです.

5.　では　自分の計算機の漢字コードは

地質調査所のネットワークにおける情報交換に　JIS 7ビット漢字コードを使う　といわれても自分の計算機で使っている漢字コードを知らないと　何をどうしたらいいのかわかりません.　ここで　計算機の漢字コードの調べ方を説明します.

(1)　まず　エディター（一太郎でもＯＫ）を使って abc あいう abc
つまり　半角の英数字　漢字　半角の英数字　という内容のファイルを例えば “codet” というような名前で作って下さい.

(2)　ユニックスワークステーションなら
od -x codet
MSDOS マシン（パソコン）なら
dump codet
として　コードの16進表現を表示させます.

(3)　表示された16進の数値が
6162 6314 a2a4 a4a4 a661 6263
であれば　その計算機の漢字コードは拡張ユニックスコードです.
6162 6382 a082 a282 a461 6263
であればシフト JIS コードです.　パソコンの多くはこのコードです.
6162 631b 2440 2422 2424 2426 1b28 4861 6263　または
6162 631b 2442 2422 2424 2426 1b28 4861 6263
ならば　JIS 7ビット漢字コードです.

ここに見られる “1b 24 40” は “ESC $ @” “1b 24 42” は “ESC $ B” そして “1b 28 48” が “ESC (J” で　漢字と英数字の切り替えのためのエスケープシークエンスです.

以上の方法で　自分の計算機の使っている漢字コードがわかります(わかるハズです).　これら以外の漢字コードを採用している計算機は少ないと思いますがないとは言えません.　そういう場合は　マニュアルを良く見るか　メーカーに問い合わせるか　少し苦労しそうです.

6.　JIS　7ビット　漢字コードとの変換は

地質調査所のネットワークにおける情報交換に JIS 7ビット漢字コードを使うということになると　よほど運が良くないかぎり自分の計算機の漢字コードのままでは　メイルも掲示板も　JUNET　も利用できません.　ニュースも読めません.　このままでは情報交換用漢字コードといいながら　絵にかいた餅になりかねません.

最近のターミナルエミュレーターソフトは　漢字コードを自動的に変換するものが多くなってきましたが　そのようなソフトをもっていなくても　漢字コードが簡単に変換できればなんとかなります.　そこで最後になりますが　漢字コードを変換するためのプログラムを紹介します.　これらのプログラムは　ユニックスのシステム言語であるＣで記述されています.　図3は以下に紹介するプログラムが　どのように漢字コードを変換するのかを示しています.

6.1.　自分の計算機が拡張ユニックス漢字コードを用いている場合のプログラム

まず紹介するのが　拡張ユニックス漢字コードの文書と　JIS 7ビット漢字コードの文書を相互に変換するためのプログラムです.

(1)　プログラムリスト—1　jtoeuc は　JIS　7ビット漢字コードで記述された文書を拡張ユニックス漢字コードの文書に変換するためのフィルタープログラムです.　これは　JIS　7ビット漢字コードで記述された文書を読むための道具です.

(2)　プログラムリスト—2　euctoj は　自分の計算機で作成した　拡張ユニックス漢字コードで記述された文書を　JIS 7ビット漢字コードの文書に変換するためのフィルタープログラムです.　これは　ネットワークに情報を乗せるのに使います.

なお　拡張ユニックス漢字コードを用いているのは Sun AS 等のワークステーションです.　VAX の漢字コード（DEC 漢字コードと呼んでいます）も　拡張ユニッ

クス漢字コードとほとんど同じものです.

6.2. 自分の計算機がシフト JIS 漢字コードを用いている場合のプログラム

次に紹介するのは シフト JIS 漢字コードの文書と JIS 7ビット漢字コードの文書を相互に変換するためのプログラムです.

(1) プログラムリスト—3 jtosj は JIS 7ビット漢字コードで記述された文書を シフト JIS 漢字コードの文書に変換するためのフィルタープログラムです.
JIS 7ビット漢字コードで記述された文書を読むために使います.

(2) プログラムリスト—4 sjtoj は シフト JIS 漢字コードで記述された文書を JIS 7ビット漢字コードの文書に変換するためのプログラムです. 自分の計算機で作った文書をネットワークに乗せるのに使います.

なお シフト JIS 漢字コードを用いているのは MSDOSパソコン SONY のワークステーション NEWS などです.

以上の4つのプログラムは いずれも標準入力から(または コマンドラインに記述したファイルから)読んで標準出力に書き出すフィルターです. ですからリダイレクション またはパイプを積極的に使ってください.
以上のプログラムには 拡張ユニックス漢字コードで記述された文書と シフト JIS 漢字コードの文書を変換するためのフィルターが含まれていません. 拡張ユニックス漢字コードの文書をファイル1とし シフト JIS 漢字コードの文書をファイル2とします. これらを変換するには 以上のプログラムとリダイレクションとパイプを使って

euctoj ファイル1 |jtosj＞ファイル2

反対向きの変換なら

sjtoj ファイル2 |jtoeuc＞ファイル2

というコマンドを打ち込めばできあがりです.

これらのプログラムはパソコンの場合ですと コンパイラーによっては動作がおかしいこともあるかもしれません. その場合は “char *argv [ ]” 以外の場所に現れる “char” を すべて “int” に置き換えて下さい. これは char 型変数のとりうる値の範囲についての解釈の不一致が原因です.

7. おわりに

地質調査所のネットワークにおける情報交換用漢字コードとして JIS 7ビット漢字を推奨するということの説明をしてきました. これについて 誤解されると困るので一言付け加えます. ここで述べた漢字コードはあくまで地質調査所のネットワークを利用した 共通のデータコミュニケーションの場の推奨コードだということです. 各人が自分の計算機で用いるのはどんなコードでもかまいませんし わかりあった同士が特有のコードで通信してもいっこうに困ったことになりません.
複数の機種の計算機の間での 情報流通を考える場合には おたがいの利益のために最低限の約束ごととして以上に説明してきた方式をとりましょう というのが私どもの真意です.

計算機ネットワークによって 情報交換が迅速におこなわれるようになれば 研究に関連したニュースは本来の速報性を取り戻し ネットワークによる速報紙(誌)の展望も開けるでしょう. また ネットワーク経由のデータ交換も可能になって 地球科学の発展に大いに寄与することでしょう. この文を読まれた皆さんが 地質調査所のネットワークを積極的に利用していただけることを期待します.

最後になりますが 筆者の JUNET のアドレスは以下のとうりです.

野呂は noro@gsjksun. gsj. junet
川勝は hitosi@gsjc3d. gsj. junet
宮崎は g0157@gsjrstn. gsj. junet

地質調査所のネットワークや この文へのご意見疑問等ありましたら上のアドレス宛にメイルをください.

参考文献

(1) JIS の情報交換用符号に関する正確な内容を知りたい方は 日本工業規格 JIS ハンドブック—情報処理 1986年 日本規格協会 5500円をご覧ください.

(2) JUNET については JUNET 利用の手引 をご覧ください. これは 通常の出版物ではないため 近代科学社 電話(03) 260—6161 に直接連絡してください. 価格は 1600円です.

(3) 言語Cについては 巷に本があふれていますが必携書は カーニハンとリッチー プログラミング言語C 共立出版でしょう.

(4) ユニックスに関する本も大変多いので 迷いますが カーニハンとパイク UNIX プログラミング環境 アスキー出版 村井ほか プロフェショナル UNIX アスキー出版等が 入門書ではありませんが役に立ちます.
(UNIX は AT & T が開発しライセンスしています.)

プログラム－1

```
/* jtoeuc.c --
                ＪＩＳ７ビット漢字コード体系の文書を
                拡張ユニックス漢字コードを含む文書に
                変換するためのフィルタープログラム */


#include        <stdio.h>

main(argc, argv)
        int     argc;
        char    *argv[];
{
        char    c;
        int     flag = 0;
        FILE    *fp, *fopen();

        if( argc == 1)          /* コマンドラインに引数がなければ、 */
                fp = stdin;     /* 標準入力から読む */

        else                    /* ファイルが指定されていれば、それから読む */
                if(( fp = fopen(argv[1], "r" )) == NULL )
                {
                        fprintf(stderr, "file %s not found¥n", argv[1]);
                        exit(1);
                }


        while(( c = getc(fp) ) != EOF)
        {
                if( c == 033 )  /* おや、エスケープシークエンスが来た */
                {
                        flag = esc_proc(fp); /* モードの切り替え */
                        continue;
                }

                if( flag == 0 )         /* 英数字モードだから、そのまま */
                        putchar(c);
                else                    /* 漢字だから、８ビット目を立てる */
                        putchar( c | 0x80 );
        }
}

int     esc_proc(fp)
        FILE    *fp;
{
        char c1 = getc(fp), c2 = getc(fp);

        if(( c1 == '(' ) && ( c2 == 'H' ))      /* 英数字:誤りだが、しばしば */
                return(0);                      /*  使われている */
        if(( c1 == '$' ) && ( c2 == '@' ))      /* 漢字:1978 */
                return(1);
        if(( c1 == '(' ) && ( c2 == 'J' ))      /* 英数字:正しい */
                return(0);
        if(( c1 == '$' ) && ( c2 == 'B' ))      /* 漢字:1983 */
                return(1);
}
```

プログラム－２

```
/* euctoj.c ―
                拡張ユニックスコードを含む文書を
                ＪＩＳ７ビット漢字体系の文書に
                変換するためのフィルタープログラム */

#include        <stdio.h>

main(argc, argv)
        int     argc;
        char    *argv[];
{
        char    c;
        int     flag = 0;
        FILE    *fp, *fopen();

        if( argc == 1 )                 /* コマンドラインアーギュメントが */
                fp = stdin;             /*  なければ標準入力から読む */

        else                            /* ファイル名があれば、それから読む */

                if(( fp = fopen(argv[1], "r")) == NULL )
                {
                        fprintf(stderr, "file %s not found¥n", argv[1]);
                        exit(1);
                }

        while(( c = getc(fp)) != EOF )
        {
                if( flag == 0 )                 /* 今、英数字モード */
                {
                        if(( c & 0x80 ) == 0)   /* 英数字なので、そのまま */
                                putchar(c);
                        else                    /* おっと、漢字だよ */
                        {
                                flag = put_esc(flag); /* エスケープシークエンス
                                                         を出力して、今から
                                                         漢字モード */
                                putchar( c & 0x7f );  /* ｅｕｃ漢字だから、
                                                         ８ビット目をマスク */
                        }
                }
                else                            /* 今は、漢字モード */
                {
                        if(( c & 0x80 ) == 0)   /* おっと、英数字が来たよ */
                        {
                                flag = put_esc(flag); /* エスケープシークエンス
                                                         を出力して、今から
                                                         英数字モード */
                                putchar(c);
                        }
                        else                    /* ｅｕｃ漢字だから、
                                                   ８ビット目をマスク */
                                putchar( c & 0x7f );
                }
        }
        if( fp != stdin ) fclose(fp);
}

int     put_esc(flag)
        int     flag;
{
        if( flag == 0 )         /* 英数字モードから、漢字モードに入る */
        {
                putchar(0x1b);
                putchar('$');
                putchar('@');

                return(1);
        }
        else                    /* 漢字モードから、英数字モードに入る */
        {
                putchar(0x1b);
                putchar('(');
                putchar('J');

                return(0);
        }
}
```

プログラム－３

```
/* jtosj.c -- JIS to SHift JIS                                    *
*             ＪＩＳ　７ビット　漢字体系の文書を                 *
*             シフトＪＩＳ　漢字体系の文書に変換するための       *
*             フィルタープログラム                               */

#include        <stdio.h>

main(argc, argv)
        int     argc;
        char    *argv[];
{
        int     c;
        int     flag = 0;
        FILE    *fp, *fopen();

        if( argc == 1)          /* コマンドラインにファイルの指定が
                                   なければ標準入力から読む              */
                fp = stdin;

        else                    /* ファイルが指定されていれば
                                   そのファイルから読む                  */
                if(( fp = fopen(argv[1], "r" )) == NULL )
                {
                        fprintf(stderr, "file %s not found¥n", argv[1]);
                        exit(1);
                }


        while(( c = getc(fp) ) != EOF)
        {
                if( c == 033 )          /* おや、エスケープが来たよ      */
                {
                        flag = esc_proc(fp);
                        continue;
                }

                if( flag == 0 )         /* 英数字なら、そのまま出力      */
                        putchar(c);
                else                    /* 漢字だから、シフトＪＩＳに    */

                        sjproc(c, fp);
        }
}

int     esc_proc(fp)            /* エスケープシークエンスの処理          */
        FILE    *fp;
{
        char c1 = getc(fp), c2 = getc(fp);

        if(( c1 == '(' ) && ( c2 == 'H' ))      /* 英数字　間違いだが    */
                return(0);                      /* よく　使われている    */

        if(( c1 == '$' ) && ( c2 == '@' ))      /* 漢字　１９７８        */
                return(1);
        if(( c1 == '(' ) && ( c2 == 'J' ))      /* 英数字　正しい        */
                return(0);
        if(( c1 == '$' ) && ( c2 == 'B' ))      /* 漢字　１９８３        */
                return(1);
}

sjproc(c, fp)                   /* ＪＩＳコードから、シフトＪＩＳコードへ */
        int     c;
        FILE    *fp;
{
        int     c2 = getc(fp);

        if(( c % 2 ) == 0 )
                c2 = c2 + 0x7d;
        else
                c2 = c2 + 0x1f;

        if( c2 >= 0x7f )
                c2 = c2 + 1;

        c = ( c - 0x21 ) / 2;

        c = c + 0x81;

        if( c > 0x9f )
                c = c + 0x40;

        putchar(c);

        putchar(c2);

}
```

プログラム－４

```
/* sjtoj.c -- Shift JIS to JIS                                    *
*                シフトＪＩＳ漢字コードの文書を                    *
*                ＪＩＳ　７ビット　漢字体系の文書に変換する       *
*                フィルタープログラム                              */

#include        <stdio.h>

main(argc, argv)
        int     argc;
        char    *argv[];
{
        int     c;
        int     flag = 0;
        FILE    *fp, *fopen();

        if( argc == 1 )         /* コマンドラインにファイル指定がなければ
                                   標準入力から読む                    */
                fp = stdin;

        else                    /* ファイルが指定されていれば、
                                   それから読む                        */

                if(( fp = fopen(argv[1], "r")) == NULL )
                {
                        fprintf(stderr, "file %s not found¥n", argv[1]);
                        exit(1);
                }

        while(( c = getc(fp)) != EOF )
        {
                if( flag == 0 )                 /* 今は、英数字モードです */
                {
                        if(( c & 0x80 ) == 0)   /* 英数字なので、そのまま */
                                putchar(c);     /* 出力します             */
                        else
                        {                       /* おっと、漢字が来た     */
                                                /* エスケープ列を出力して
                                                   漢字モードへ           */
                                flag = put_esc(flag);
                                sjproc(c, fp);  /* シフトＪＩＳから
                                                   ＪＩＳに変換           */
                        }
                }
                else                            /* 今は、漢字モードです   */
                {
                        if(( c & 0x80 ) == 0)   /* おや、英数字が来た     */
                        {                       /* エスケープ列を出力して
                                                   英数字モードへ         */
                                flag = put_esc(flag);
                                putchar(c);     /* 文字は、そのまま       */
                        }
                        else                    /* 漢字が来たので         */
                                sjproc(c, fp);  /* シフトＪＩＳから       */
                                                /* ＪＩＳに変換           */
                }
        }
        if( fp != stdin ) fclose(fp);
}

int     put_esc(flag)
        int     flag;
{
        if( flag == 0 )         /* 今から漢字、というエスケープシークエンス */
        {
                putchar(0x1b);
                putchar('$');
                putchar('@');
                return(1);
        }
        else                    /* 英数字の、エスケープシークエンス       */
        {
                putchar(0x1b);
                putchar('(');
                putchar('J');

                return(0);
        }
}

sjproc(fst, fp)         /* シフトＪＩＳから、ＪＩＳへの変換               */
        int     fst;
        FILE    *fp;
{
        int     sec = getc(fp);

        if( fst <= 0x9f )
                fst = fst - 0x71;
        else
                fst = fst - 0xb1;

        fst = 2 * fst + 1;

        if( sec > 0x7f )
                sec = sec -1;

        if( sec >= 0x9e )
        {
                sec = sec - 0x7d;
                fst = fst + 1;
        }
        else if( sec < 0x9e )
                sec = sec - 0x1f;

        putchar( fst );
        putchar( sec );
}
```