AIX 6

AIX日本語処理機能ユーザーズ・ガイド

SC88-0428-03

注意！

本書および本製品をお使いになるまえに、まず A-1 ページの『特記事項』に記載されている説明をお読みください。

第 4 版　2009 年 10 月

■ はじめに

本書は、AIX 6 で使用される AIX 日本語処理機能について解説したものです。

■ 本書の構成

本書は、次の８つの章および付録から構成されます。

第１章 日本語入力機能と表示
AIX日本語処理機能の使用方法について解説してあります。

第２章 ユーザー辞書ユーティリティー
ユーザー辞書ユーティリティーの使用方法について解説してあります。

第３章 日本語プリンター・サポート
日本語プリンターの使用方法について解説してあります。

第４章 共通デスクトップ環境での日本語入出力
共通デスクトップ環境(CDE)での日本語の使用方法について解説してあります。

第５章 日本語GUIアプリケーションの作成
Motif、X Window Systemでの日本語アプリケーションの作成方法について解説してあります。

第６章 コード・コンバージョン・サポート
iconvコンバーターの使用方法について解説してあります。

第７章 日本語ユニコード・サポート
Unicodeの日本語ロケールJA_JPについて解説してあります。

第８章 JISX0213サポート
JIS第3水準および第4水準を利用するための環境について解説してあります。

付録A コンパイラーの日本語サポート
B 日本語aixtermに対する制御コード
C AIX日本語処理機能の制限事項
D JIS8ビット文字コード表
E ローマ字入力ーかな字対応表
F 記号読み入力の一覧表
G ターミナル・エミュレーション
H プログラム・リスト immsample.c
I プログラム・リスト ximsample.c
J プログラム・リスト mrsample.c
K プログラム・リスト losample.c

■ 関連マニュアル

本書に関連するマニュアルには以下のものがあります。

- AIX フォント・ユーティリティー ユーザーズ・ガイド	SC88-0002
- AIX 5L 日本語コード一覧表	SC88-0427
- AIX 日本語環境拡張キット　バージョン2.4　ユーザーズ・ガイド	SC88-0453
- Wnn8 for AIX ユーザーズ・マニュアル	SC88-0454

また、以下の URL から AIX に関する技術情報ソースを参照できます。

http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.doc/doc/base/aixparent.htm

■ 目次

このページは空白です。

1. 日本語入力機能と表示

■ 目次

1.1 ＡＩＸ日本語入力機能の予備知識

本節では、AIX 日本語入力機能を使用する上での予備知識について解説します。

1.1.1 位置づけ

AIX 日本語入力機能とは、AIX 上で動く多くの適用業務プログラムに日本語を入力することを可能にするプログラムです。
通常、キーボードから入力された文字は図１- a に示されたように、そのまま適用業務プログラムに送られます。

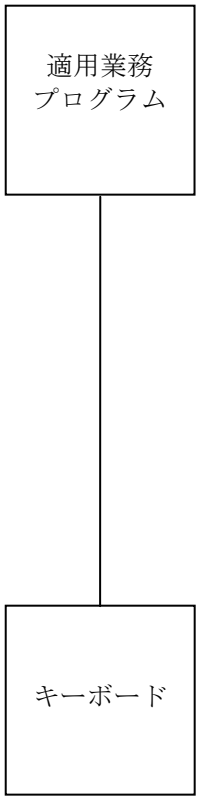


図１- a

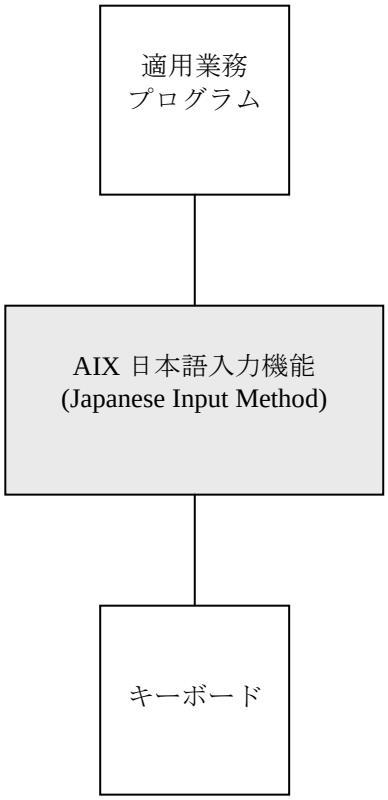


図１- b

AIX 日本語入力機能は、このキーボードと適用業務プログラムの間に位置し、適用プログラムに対し日本語の入力を可能にします。（図１- b）

例えば、ユーザーがひらがなを入力した場合、これらのひらがなは適用業務プログラムに直接渡されず、一旦 AIX 日本語入力機能によりエコーバック（画面に表示）されます。このエコー・バックされている文字列に対して、ユーザーは変換や文字の消去などの編集機能を使用することができます。希望の日本語文字列が得られてユーザーが Enter キーを押すと、これらのエコー・バックされていた文字列が一度に適用業務プログラムに送られます。

AIX 日本語入力機能は、日本語を入力する上で必要な多くの編集機能を提供します。例えば『変換』や『文字の消去』などです。これらの編集機能が使用できる範囲は画面上では下線によって示されます。

注）AIX 以外のマシン上で AIX のリモート端末を使用する場合は、そのマシン上で稼動している入力機能が使われます。Wnn8日本語入力機能は、AIX日本語環境拡張キットで提供されます。Wnn8日本語入力機能の使用方法に関しては、以下のマニュアルを参照してください。

「AIX日本語環境拡張キット バージョン2.4 ユーザーズ・ガイド」(SC88-0453)
「Wnn8 for AIX ユーザーズ・マニュアル」(SC88-0454)

1.1.2 入力フィールド

入力フィールドとは、日本語テキストの入力、編集を行う領域です。日本語端末エミュレーターでは、入力フィールドは画面全体です。（ただし、ウィンドウの下端のステータス・ラインは除きます。）

ユーザが入力した文字は、通常入力フィールドにエコー・バックされます。入力された文字が AIX 日本語入力機能の編集対象文字となる場合（例えばひらがなを入力した場合）エコー・バックされた文字は下線によって示されます。

注）「AIX 日本語入力機能の編集対象文字」とは AIX 日本語入力機能によって変換や削除などの編集操作が可能な文字のことを言います。ひらがなは、通常、編集対象文字となります。

AIX 日本語入力機能によって入力フィールドにエコー・バックされた文字は下線によって示されますが、これらの文字は更に反転表示されることがあります。この反転表示はこれらの文字が AIX 日本語入力機能の編集機能の内、特に変換操作ができる可能性を示します。これらの反転表示されている文字のことを変換対象文字と言います。逆に言うと、反転表示されていない文字は 変換 キーを押しても変換されません。

1.1.3 入力モード

日本語キーボードの各キーには、いくつかの文字が割り当てられています。どの文字が入力されるかはキーボードの状態で決まります。このキーボードの状態のことを入力モードと呼びます。

次に示す３種類の入力モードがあります。

英数字／ひらがな／カタカナ　入力モード

これは、現在入力される文字の種類を表すモードです。文字の種類としては「英数字」、「ひらがな」および「カタカナ」があります。
英数 キー、ひらがな キー、および カタカナ キーを押すとそれぞれのモードになります。

全角／半角　入力モード

これは、入力される文字の大きさが全角か半角かを示すモードです。
全角／半角 キーを押すと、このモードが交互に切り替わります。

ローマ字　入力モード

ローマ字入力モードでは、「ひらがな」または「カタカナ」を入力する時にそのかな文字に対応したローマ字で入力することができます。
例えば、「か」という文字を入力する時には、「ka」と入力します。
ローマ字 キー（ Alt キーを押しながら ひらがな キーを押す）を押すと、このモードが交互に切り替わります。

これら３種類のモードの組み合わせで、現在どのような文字が入力されるかが決まります。これらの入力モードが現在どのようになっているかが端末エミュレータのステータスラインに表示されます。

英数／ひらがな／カタカナ　入力モードは、

「英数」、「かな」または「カナ」

と表示されます。

全角／半角　入力モードは、

「全角」、または「半角」

と表示されます。

ローマ字　入力モードの時には、

「R」

が表示されます。

例えば、「ローマ字、全角、カタカナ」入力モードの時には次のように表示されます。

カナ　全角　R

ローマ字入力に関する注意点

- 『ん』の扱い
  『ん』は『な行』の文字との混乱を避けるために”nn”と入力します。

  但し、後ろに子音（n と y を除く）が続く場合に、スペース キー、 変換 キーまたは

  無変換 キーを押すと”n”が『ん』と解釈されます。

- 促音の扱い
  つまる音『っ』は、その後ろに続く子音を重ねることで入力します。

  例）　gakkou　→　がっこう

- よう音の扱い
  『ゃ』『ょ』などの小さく書かれる音は、読みの通りに入力すれば入力できます。また”x”を用いて入力することもできます。

  例）kyou　→　きょう

  xya　→　ゃ

  xyo　→　ょ

- その他
  『を』は”wo”と入力します。

- 入力途中のアルファベット（子音）を消す場合は、 ← （後退）キーを押します。
  この場合、入力途中のアルファベットが消えるだけで、画面上の変化はありません。

  詳しくは付録の『ローマ字入力ーかな文字対応表』を参照してください。

1.1.4 変換モード

AIX 日本語入力機能のかな漢字変換には、次に示す３種類の変換モードがあります。

- 連文節変換
  連文節変換とは、いくつもの文節を一度に変換する方法です。長い文章でも高い精度で変換することができます。

  例）かなからかんじにへんかんします　→　かなから漢字に変換します

  　　ぶんしょうたんいにへんかんします　→　文章単位に変換します

- 単文節変換
  単文節変換とは、１つの文節ごとに変換する方法です。

  例）わたしは　→　私は

  　　かんじを　→　漢字を

- 複合語変換
  複合語とは、『情報処理学会』や『電子回路基板』等のように、いくつかの熟語が集まってできた語句です。複合語変換モードでは、このような複合語を高い精度で変換することができます。部品表や商品名等を入力する場合に便利です。

  同じ読みを入力しても変換モードにより変換結果が異なることがあります。

  例）こうしゅうは　→　講習は（連文節変換、単文節変換）

  　　こうしゅうは　→　高周波（複合語変換）

変換モードを変えるには Alt キーを押しながら カタカナ （ Shift + ひらがな ）キーを押します。すると次のようなメニューを表示しますので、そこで希望する変換モードに対応した番号を入力してください。現在の変換モードは、メニュー上で反転表示されます。

変換モード：
１．先読み連文節
２．一括連文節
３．単文節
４．複合語

注）表示されたメニューには、『先読み連文節』と『一括連文節』の２種類の連文節変換モードがあります。
『先読み連文節』では、『読み』が１文字入力されるごとに日本語解析が行われます。
変換 キーを押すと、すでに解析された結果をもとに変換しますので、それまでに入力された『読み』の日本語解析が一括して行われます。
『先読み連文節』と『一括連文節』で『読み』の入力方法に違いはありません。

1.1.5 学習機能

AIX 日本語入力機能は、『読み』に対する最新の変換結果を記憶します。この機能を学習機能と言います。この為同じ『読み』を入力しても違う結果が得られることがあります。

例えば『かんじ』という『読み』を『幹事』という漢字に変換しようとして 変換 キーを押したが、『幹事』という漢字は得られず『漢字』という漢字が得られた場合を考えてください。この場合、更に何度か 変換 キーを押すと『幹事』という漢字が得られます。この後再び『かんじ』という『読み』を入力して 変換 キーを押すと、今度は『幹事』という漢字が最初の変換で得られます。

端末エミュレーターは終了すると学習結果をユーザー辞書に保存します。但し、プロファイルに指定することにより学習結果を保存しないようにすることもできます。プロファイルの指定の仕方は『日本語処理機能のカストマイズ』の解説を参照してください。

1.1.6 日本語処理機能のカストマイズ

日本語処理機能プロファイル

プロファイルを作成することにより、日本語処理機能をカストマイズすることができます。
プロファイルの各行は、次に示すような形でなければなりません。

　オプション名：オプションの値

但し"#"で始まる行はコメントとして扱われます。

プロファイルは次に示す方法で指定することができます。
環境変数 JIMPROFILE にプロファイルのパス名が指定されていれば、そのプロファイルが使われます。
環境変数 JIMPROFILE の指定がなく、ユーザーのホーム・ディレクトリに.jimrc というファイル名でプロファイルがあれば、そのプロファイルが使われます。
更に.jimrc がない場合は/usr/lpp/jls/defaults/jimrc が使われます。

initrkc　ローマ字入力モードの初期値を設定します。

initrkc:off	ローマ字入力モードオフ（省略時）
initrkc:on	ローマ字入力モードオン

initsize　文字の大きさ（全角／半角）の初期値を指定します。

initsize:single	半角文字入力モード（省略時）
initsize:double	全角文字入力モード

initchar　文字の種類（英数字、カタカナ、ひらがな）の初期値を指定します。

initchar:alphanum	英数字入力モード（省略時）
initchar:katakana	カタカナ入力モード
initchar:hiragana	ひらがな入力モード

conversion 変換モードの初期値を設定します。

conversion:look-ahead	先読み連文節変換
conversion:mphrase	一括連文節変換（省略時）
conversion:sphrase	単文節変換
conversion:word	複合語変換

kjbeep　キー入力エラー等によって鳴るビープ音の初期値を設定します。

kjbeep:on	ビープ・オン（省略時）
kjbeep:off	ビープ・オフ

learning　変換の学習結果を辞書へ保存するかどうかを指定します。
learning:on　学習機能オン（省略時）
learning:off　学習機能オフ

alphanum　英数字も漢字変換の対象にするかどうかを指定します。
alphanum:on　変換対象にする（省略時）
alphanum:off　変換対象にしない

kuten　区点番号として使うコードを指定します。
kuten:JIS83　83 年度 JISX0208（省略時）
kuten:JIS78　78 年度 JISX0208

upload　日本語処理機能動作中に、必要に応じてユーザー辞書ファイルを読み直します。これによってユーザの登録した語句がすぐに他のプログラムに反映されます。
upload:on　日本語処理機能動作中に、もしユーザー辞書ファイルが他のプロセスにより更新されたらなら、その日本語処理機能は自動的にユーザー辞書ファイルを読み直します。
upload:off　日本語処理機能は、それが始動した時に読み込んだユーザー辞書ファイルを使用し続けます。

注）以下の特殊な環境下では、かな漢字変換の速度が遅くなる恐れがあります。その際には”upload:off”に設定して下さい。

ユーザー辞書ファイルを NFS マウントで使用しているが、そのシステムのネットワーク状態がトラブルの為に非常に悪い時。

numberinput　漢字コード入力モードの初期値を設定します。
numberinput:JISkuten　JIS 区点番号入力（省略時）
numberinput:IBMkanji　漢字番号入力

modereset　Esc キーが日本語処理機能を初期化するかどうかを指定します。
modereset:on　Esc キーを押すと、日本語処理機能は初期化されます。これにより強制的にその入力モードは”英数・半角”になります。この機能は特に vi エディターで日本語テキストファイルを作成するユーザーにとっては有効です。
modereset:off　Esc キーを押しても、何も起こりません。（省略時）

singlequote　全角シングルクォート記号に対するコードポイントを設定します。
singlequote : 8166　JISのコードポイント（IBM-943での省略時）
singlequote : fa56　IBM選定文字のコードポイント（IBM-eucJPでの省略時）

doublequote　全角ダブルクォート記号に対するコードポイントを設定します。
doublequote : 8168　JISのコードポイント（IBM-943での省略時）
doublequote : fa57　IBM選定文字のコードポイント（IBM-eucJPでの省略時）

shmatnum　AIX日本語入力機能が使用する共有メモリーの数を指定します。ただし、環境変数JIMSHMATNUMに値が指定されていれば、その値が使われます。
shmatnum : N　N個のセグメントを使用（省略時の値は1）

/usr/lpp/jls/defaults/jimrcを各ユーザーのホーム・ディレクトリーに.jimrcとコピーして使用して下さい。

日本語処理機能キー

xmodmapコマンドを使用することにより、変換 キー等の日本語処理機能用に使われるキーを他のキーに割り当てることができます。この機能を使用することにより、例えば英語キーボードで日本語を入力することも可能です。xmodmapコマンドについては下記資料も参照してください。

AIX Version 6.1 Commands Reference, Volume 6　SC23-5248

xmodmapコマンドではkeysymと呼ばれるキーボード上のキーのシンボル名を使用します。
以下の keysym が日本語処理機能に使われる keysym です。

Kanji	：	変換キー
Muhenkan	：	無変換キー
Romaji	：	ローマ字キー
Hiragana	：	ひらがなキー
Katakana	：	カタカナキー
ZenkakuHankaku	：	全角／半角キー
Zenkaku	：	全角キー
Hankaku	：	半角キー
Eisu_toggle	：	英数キー

以下の keysym は AIX に固有な keysym です。
AIX 以外の xmodmap コマンドを使用する場合はカッコ内に示された 16 進数を keysym の代わりに使用して下さい。

ZenKouho	(0x1800ff01)	：	全候補キー
KanjiBangou	(0x1800ff02)	：	漢字番号キー
HenkanMenu	(0x1800ff03)	：	変換モード指定キー
MaeKouho	(0x1800ff04)	：	前候補キー
BunsetsuYomi	(0x1800ff05)	：	文節読みキー
LeftDouble	(0x1800ff06)	：	倍速カーソル移動キー（左）
RightDouble	(0x1800ff07)	：	倍速カーソル移動キー（右）
ErInput	(0x1800ff0a)	：	入力キャンセル・キー
Reset	(0x1800ff0b)	：	リセット・キー

/usr/include/X11/aix_keysym.h ファイルの中に上記の keysym が定義されています。
aix_keysym.h ファイルには他に LeftPhrase と RightPhrase が日本語処理機能用の keysym として定義されていますが、これらは使用されていません。）

以下に日本語処理機能用キーをファンクション・キーに割り当てる例を示します。

1. エディター等を使用して以下のようなファイルを作ります。

```
$ vi mykeys
                （編集）
$ cat mykeys

keysym F1 = Eisu_toggle Katakana
keysym F2 = Hiragana Romaji
keysym F3 = Zenkaku_Hankaku HenkanMenu
keysym F4 = Kanji MaeKouho
keysym F5 = Muhenkan BunsetsuYomi
keysym F6 = ZenKouho KanjiBangou
```

2. xmodmap コマンドを実行します。

```
$ xmodmap mykeys
```

なお、キーの割り当ては xmodmap コマンド実行後、新たに開いた端末エミュレーターで使用可能になります。
Shift キーを押しながら、ファンクション・キーF1 を押すと、カタカナモードになります。

システム上のキーの割り当ては、/usr/lpp/Xll/defaults/xmodmap/$LANG/keyboard ファイル上に設定されています。
各ユーザーごとに keyboard ファイルを指定する場合は、/usr/lpp/Xll/defaults/xmodmap/$LANG/keyboard ファイルを各ユーザーのディレクトリーにコピーしてから上記を加えて xmodmap コマンドを実行して下さい。

1.1.7 日本語辞書

日本語辞書には次の３種類があります。

1. システム辞書
   一般的な語句が登録されている辞書です。

2. ユーザー辞書
   ユーザー固有の用語を登録する辞書です。

3. 付属語辞書
   付属語を、ひらがなか漢字のどちらで表示するのかを記憶している辞書です。

   例）（漢字で表示する場合）　　（ひらがなで表示する場合）
   暑中お見舞い申し上げます。　　申しあげます
   筆記用具を持参して下さい。　　ください

これらの辞書は/usr/lpp/jls/dict ディレクトリーに存在し、通常これらが使用されます。使用する辞書を変更するには次のようにします。

① システム辞書

IBM が提供するシステム辞書は次の３つです。

システム辞書	内容	読み
ベース辞書 /usr/lpp/jls/dict/ibmbase	名詞、動詞、単漢字等基本となる辞書。 約 85,000 語。 例）報道、伝わる	全角ひらがな ほうどう、つたわる
国名コード辞書 /usr/lpp/jls/dict/ibmcnnc	JIS 国名コードに対応した国名、および最新の国名を追加して登録。 例）日本国、イタリア共和国	全角英字（国名コード） （読みのサンプルとして「JIS ハンドブック情報処理用語・コード編、国名コード X 0304-1988」の２文字、３文字コードを準備。大文字でも小文字でも使用可。） JPN,　ITA
郵便番号辞書 /usr/lpp/jls/dict/ibmzipc	３桁郵便番号に対応した地名 例）東京都豊島区 　　岩手県盛岡市	全角数字（郵便番号） （読みのサンプルとして県庁所在地および東京 23 区の郵便番号（3 桁）を準備。） １７０ ０２０

これらのシステム辞書は、以下の方法で AIX 日本語処理機能において使用することができます。

環境変数 JIMMULDICT にシステム辞書名（パス名）が指定されていれば、その辞書が使われます。

例）任意のディレクトリー上にあるシステム辞書、例えば”/usr/jpn/ibmbase”を aixterm 上で使う。

```
$ JIMMULDICT=/usr/jpn/ibmbase
$ export JIMMULDICT
$ aixterm –lang ja_JP &
```

例）IBM が提供しているベース辞書、国名コード辞書、郵便番号辞書を aixterm 上で同時に使う。

```
$ JIMMULDICT=/usr/lpp/jls/dict/ibmbase:/usr/lpp/jls/dict/ibmcnnc:/usr/lpp/jls/dict/ibmzipc
$ export JIMMULDICT
$ aixterm –lang ja_JP &
```

「：」でシステム辞書名を区切ることにより、システム辞書を最大 16 個まで指定できます。

指定したシステム辞書の中に、１つでも不正なものがあった場合には、
/usr/lpp/jls/dict/ibmbase だけが使われます。

環境変数 JIMMULDICT が指定されていなければ、/usr/lpp/jls/dict/ibmbase が使われます。

AIX 日本語入力機能で使用されるシステム辞書ファイルは、以下のシステムで使用されているマルチシステム辞書ファイルと、互換性があります。

IBM OS/2-J バージョン 1.3 以上
連文節変換プログラム バージョン 3.0（IBM DOS バージョン J4.0/V 以上）

その為、上記システムで使われているマルチシステム辞書を AIX システム上で使用することが可能です。（ユーザー辞書、付属語を除く）

注）AIX バージョン 3.2.2 以前でサポートしていた環境変数 JIMSYSDICT は、サポートされません。
AIX バージョン 3.2.2 以前で提供していたシステム辞書、/usr/lpp/jls/dict/sysdict は使用できません。ユーザー辞書は従来通り使用できます。

② ユーザー辞書

環境変数 JIMUSRDICT にユーザー辞書名が指定されていれば、その辞書が使われます。
環境変数 JIMUSRDICT の指定がなくユーザーのホーム・ディレクトリーに.usrdict というファイル名でユーザー辞書があれば、その結果が使われます。
ユーザー辞書がホーム・ディレクトリーにも無い場合には /usr/lpp/jls/dict/usrdict が使われます。

③ 付属語辞書

環境変数 JIMADJDICT に付属語辞書名が指定されていれば、その辞書が使われます。
環境変数 JIMADJDICT の指定がなく、ユーザーのホーム・ディレクトリーに.adjdict というファイル名で付属語辞書があれば、それが使われます。
付属語辞書がホーム・ディレクトリーにも無い場合には、/usr/lpp/jls/dict/adjdict が使われます。

注）ユーザー辞書および付属語には学習効果が保存されます。（ただし、保存しないように指定した場合を除く）
従って、これらの辞書には書き込み許可が与えられていなければなりません。これらの辞書は、各ユーザーのホームディレクトリーにコピーして使用することをお薦めします。

1.2 ＡＩＸ日本語入力機能

本節では、AIX 日本語入力機能を使用して日本語を入力する方法について具体例を示しながら解説します。

1.2.1 漢字に変換する

漢字を入力する為には、その漢字の読みを入力して 変換 キーを押します。

[操作例]

1. はじめに『ひらがな』入力モードにしてください。

   ローマ字による入力をする場合には ローマ字 キー（ Alt キーを押しながら ひらがな キーを押す）、また連文節変換モードになっていない場合は、 Alt キーを押しながら
   カタカナ （ Shift + ひらがな ）キーを押して連文節変換モードを選んで下さい。

2. 『かなからかんじにへんかんします』と入力して下さい。

   かなからかんじにへんかんします __

   正しく入力されていることを確認してください。この時点で間違いに気付いた場合は Delete キー、
   Back Space キー、 → キー、 ← キーで修正することができます。

3. 変換 キーを押して下さい。

   かなから漢字に変換します __

場合によっては変換結果がこの例と異なることがあります。希望の漢字が得られなかった場合には再変換が必要です。

[操作例]

1. 前の例と同じ『読み』を入力して 変換 キーを押したが、次のように異なる結果が得られた場合。

   かなから幹事に変換します __

2. ← キーを押してカーソルを『幹事に』の位置に移動して下さい。

   かなから幹事に変換します __

3. この状態で、もう一度 変換 キーを押して下さい。

   かなから漢字に変換します __

4. それでも『漢字』に変換されない場合は、更に何度か 変換 キーを押してください。次回からは学習機能により希望の漢字がすぐに得られるようになります。

上の例でも見られるように、カーソルを移動すると反転表示される部分が変わります。反転表示されている部分のことを『変換対象』と言い、この部分が 変換 キーを押すことにより変換されることを示します。カーソルを移動しても反転表示されない部分は、再変換できません。 変換 キーを押すと AIX 日本語入力機能はその時の学習状況等から判断して可能性の高い候補から順に表示していきます。シフトキーを押しながら 変換 キーを押すと漢字候補を逆順に表示します。但し、この操作は一度 変換 キーを押した後でないと働きません。

希望の漢字が得られたら、 Enter キーを押してください。すると下線および反転が消え、現在表示されている漢字に確定します。一度確定すると再変換はできなくなります。

1.2.2 読みのままにしておく

時には変換する必要のないことがあります。『読み』のままにしておくには 無変換 キーを使います。

[操作例]

1. 次のような文章を作ってみましょう。

   「漢字」は「かんじ」と読みます

2. まず『「かんじ」は』と入力して 変換 キーを押して下さい。

   「漢字」は ＿

3. 次に『「かんじ」と』と入力して下さい。

   「漢字」は「かんじ」と ＿

4. ここで 無変換 キーを押して下さい。反転表示が消えます。

   「漢字」は「かんじ」と ＿

5. 次に『よみます』と入力して 変換 キーを押して下さい。

   「漢字」は「かんじ」と読みます ＿

無変換 キーは、この例のように『読み』のまま残したい時に使います。
また、 無変換 キーを用いることにより『ひらがな』を『カタカナ』に変換することもできます。

[操作例]

1. ひらがな入力モードで『ぷろぐらむ』と入力して下さい。

   ぷろぐらむ

2. この状態のまま、 無変換 キーを押してください。

   ぷろぐらむ

3. ここでもう一度 無変換 キーを押してください。『カタカナ』に変換されます。

   プログラム

4. もう一度 無変換 キーを押すと、はじめの状態に戻ります。

   ぷろぐらむ

無変換 キーを更に押すと、前の例の２）、３）、４）を繰り返します。但し、この機能は『読み』を入力した直後のみ可能です。

無変換 キーには、もう１つの異なった機能があります。この機能については次の節で説明します。

1.2.3 漢字を読みに戻す

『読み』を誤って変換してしまった場合や文節が間違って区切られてしまった場合等には、『読み』に戻すという作業が必要になります。 無変換 キーによる方法と、 文節読み キー（ Alt キーを押しながら 無変換 キーを押す）による方法があります。

文節読み キーは１つの文節のみを『読み』に戻します。

[操作例]

1. 『「かんじ」は「かんじ」とよみます』と入力して 変換 キーを押して下さい。

   「漢字」は「漢字」と読みます

2. カーソルを２番目の『漢字』の位置に移動して下さい。

   「漢字」は「漢字」と読みます

3. ここで 文節読み キーを押して下さい。

   「漢字」は「かんじ」と読みます

4. Enter キーを押すと確定します。

無変換 キーは現在のカーソル位置より右側部分を全て『読み』に戻します。これは AIX 日本語入力機能が誤って文節を区切ってしまった場合に有効です。

[操作例]

1. 『ここではきものをぬいでください。』と入力し 変換 キーを押して下さい。

   ここでは着物を脱いで下さい。

2. ← キーを用いてカーソルを先頭位置まで移動して 無変換 キーを押して下さい。

   ここではきものをぬいでください。

3. → キーを用いて正しい文節のみを反転表示させて下さい。

   ここではきものをぬいでください。

4. 無変換 キーを押して下さい。

   ここではきものをぬいでください。

5. 次に → キーを用いてカーソルを最後の『い』の位置まで移動して下さい。

   ここではきものをぬいでください。

6. 変換 キーを押して下さい。

   ここで履き物を脱いで下さい。

1.2.4 カーソル移動

すでに操作例にも出てきたように、カーソルの移動は [→] および [←] キーで行います。これらのキーは、カーソルを１文字分右または左に移動します。

カーソルをより速く移動させたい場合は、 [Alt] キーを押しながら [→] または [←] キーを押すと右または左に一度に２文字分ずつ移動することができます。

端末エミュレーターでは、カーソルを下線が引かれている範囲外に移動させることはできません。（もちろん、下線が無い場合には入力フィールド内で自由に移動させることができます。）

また、端末エミュレーターでは下線の引かれている範囲が複数行にまたがる場合があります。この場合には [↑] キーまたは [↓] キーを押すことにより上または下の行に移動させることもできます。但し、下線の引かれている範囲外へ移動させることはできません。

1.2.5 文字の訂正

間違った文字を入力した場合や、作成した文章を訂正する場合には、文字を消す必要が生じます。AIX 日本語入力機能では、文字を消す方法として４種類のキーが用意されています。

[←] （後退）キーは現在のカーソル位置の左側１文字を消します。但し、ローマ字入力をしている場合で、子音を入力した直後の [←] （後退）キーは、今入力した子音を消します。この場合は画面上の文字は消えません。
[削除] キーは、現在のカーソル位置の１文字を消します。

[End] キーは、現在のカーソル位置の文字とそれより右側にある全ての文字を消します。

[Alt] キーを押しながら [End] キーを押すと、AIX 日本語入力機能によって処理中の全ての文字が消されます。

1.2.6 全候補変換

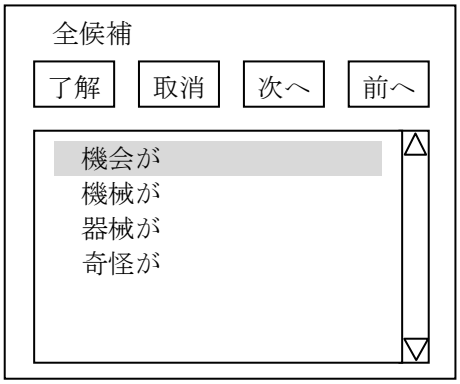
[変換] キーを押してもなかなか希望する候補が得られない場合に便利な方法があります。
[全候補] キー（ [Alt] キーを押しながら [変換] キーを押す）を押すと変換候補の一覧表を表示することができます。

[操作例]

1. 『きかいが』と入力し [変換] キーを押して下さい。

   機会が

2. [全候補] キー（ [Alt] キーを押しながら [変換] キーを押す）を押して下さい。次のようなウィンドウが現れ、その中に候補が表示されます。（候補がない場合には、このウィンドウは表示されません。）



3. ここで希望する候補をマウスで選択し、クリックします。あるいは、希望する候補を ↑ キーまたは ↓ キーを使用して選び Enter キーを押します。その候補が入力フィールドに表示され、全候補のウィンドウは消えます。例えば『器械が』を選んで Enter キーを押すと次のようになります。

   器械が__

   候補が 10 個以上あり、1 度に表示できない場合には、全候補ウィンドウ内のスクロールバーをマウスで操作するか PageDown キーを押すと、次の候補のリストが表示されます。
   また PageUp キーを押すと、前の候補リストが表示されます。全候補ウィンドウ内の「取消」ボタンをマウスでクリックするか、 Reset キー（ Ctrl キーを押しながら Esc キーを押す）を押すと、全候補を表示する前の状態に戻ります。

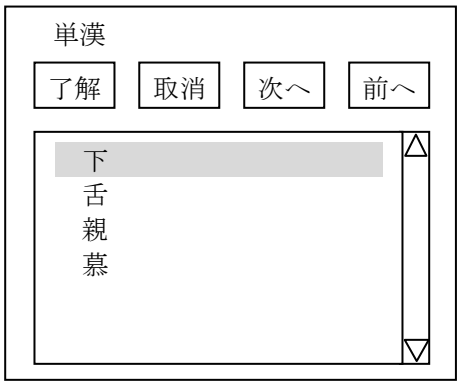
1.2.7 単漢字変換

文節単位の変換で、希望の漢字が得られない場合に有効な方法として単漢字変換という方法があります。

1. 『した』の単漢字変換の例を示します。『した』と入力して下さい。

   した __

2. 次に 全候補 キー（ Alt キーを押しながら 変換 キーを押す）を押して下さい。次のようなウィンドウが現れ、単漢字の候補の一覧表が表示されます。（候補がない場合には、このウィンドウは表示されません。）



3. ここで希望する候補をマウスで選択しクリックします。あるいは、希望する候補を ↑ キーまたは ↓ キーを使用して選び Enter キーを押します。その候補が入力フィールドに表示され、全候補のウィンドウは消えます。例えば『親』を選んで Enter キーを押すと次のようになります。

   親 __

   この時反転表示が消え、再変換できなくなります。

   候補が 10 個以上あり、1 度に表示できない場合には全候補ウィンドウ内のスクロールバーをマウスで操作するか PageUp キーを押すと次の候補のリストが表示されます。

   また PageUp キーを押すと、前の候補リストが表示されます。全候補ウィンドウ内の「取消」ボタンをマウスでクリックするか、 Reset キー（ Ctrl キーを押しながら Esc キーを押す）を押すと、全候補を表示する前の状態に戻ります。

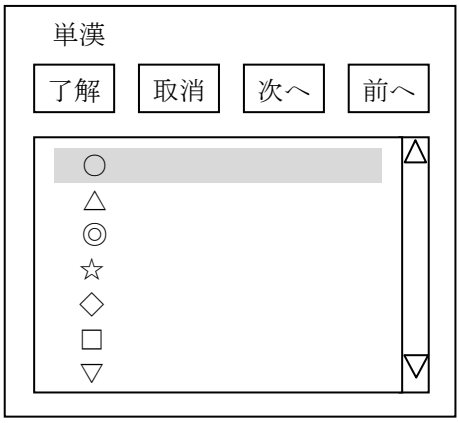
   単漢字変換は記号を入力したい時にも便利です。付録の『記号読み入力の一覧表』を参照してください。この表に示されている記号は、特別の名前で登録されていますので、この登録名で単漢字変換を行うと簡単に記号を得ることができます。

[操作例]

1. 記号『☆』を得る例を示します。『しろきごう』と入力して下さい。

   しろきごう _

2. 次に 全候補 キー（ Alt キーを押しながら 変換 キーを押す）を押して下さい。次のようなウィンドウが表示されます。（候補がない場合には、このウィンドウは表示されません。）



3. 『☆』を選択します。

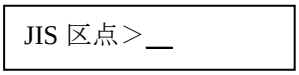
   ☆

1.2.8 ＪＩＳ区点番号入力

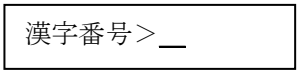
漢字コードの入力方法として、JIS 区点番号入力による方法、または漢字番号入力による方法があります。

番号キー（ Alt キーを押しながら ¥ キーを押す）を押すと、漢字コード入力の為のウィンドウが表示されます。

JIS 区点番号入力用



漢字番号入力用



番号キーを連続して押すことによりこの２つのウィンドウが切り替わります。
入力したい漢字コードの種類に応じて使い分けてください。

例）

	コード	表示文字
JIS 区点番号入力の場合	4424	霧
漢字番号入力の場合	3656	霧

注）
日本語処理機能プロファイルをカストマイズすることにより、’83 年版の JIS 区点コード’ を使うか、’78 年版の JIS 区点コード’ を使うかが決定されます。詳しくは”1.1.6 日本語処理機能のカストマイズ” を参照してください。

漢字コードは「AIX 5L 日本語コード一覧表」(SC88-0427)を参照して下さい。

1.2.9 単語登録

ユーザー辞書ユーティリティーを使用せず、ユーザー固有の語句をユーザー辞書に簡単に登録することができます。登録キー（ Alt キーを押しながら全角／半角キーを押す）を押すと、次のような単語登録ウィンドウが現れ、そこから語句と読みを入力することができます。

```
  単語登録      辞書：/u/user1/.usrdict
語句[_                                ]
読み[                                 ]
Enter=登録   ESC=終了
```

単語を登録する辞書としては、現在日本語入力機能が使用している辞書が使用され、そのファイル名が一行目に表示されます。語句・読みの文字数／使用できる文字の種類の制限はユーザー辞書ユーティリティーを使用した場合と同じですので、ユーザー辞書ユーティリティーの項を参照してください。

[操作例]

読み「どうぶつ」を語句「アフリカ象」で登録したい時

1. 未確定な文字が無い（反転・下線表示の文字が無い）状態で登録キー（ Alt キーを押しながら全角／半角キーを押す）を押すと、単語登録ウィンドウが表示されます。

```
  単語登録      辞書：/u/user1/.usrdict
語句[_                                ]
読み[                                 ]
Enter=登録   ESC=終了
```

2. 最初はカーソルが語句の先頭にあるので、日本語入力機能を使って「アフリカ象」と入力します。

```
  単語登録      辞書：/u/user1/.usrdict
語句[アフリカ象 _                     ]
読み[                                 ]
Enter=登録   ESC=終了
```

3. 語句の文字列が確定した（反転・下線表示の文字が無い）状態で Enter キー、 tab キーまたは ↓ キーを押すとカーソルが読みの先頭に移るので、「どうぶつ」と入力します。

```
  単語登録      辞書：/u/user1/.usrdict
語句[アフリカ象                       ]
読み[どうぶつ_                        ]
Enter=登録   ESC=終了
```

4. 読みの文字列が確定した状態で Enter キーを押すと、語句がユーザー辞書に登録され、「登録されました。」というメッセージが表示されます。

```
  単語登録      辞書：/u/user1/.usrdict
語句[_                                ]
読み[                                 ]
登録されました。    ESC=終了
```

5. ESC キーを押すと単語登録ウィンドウが消え、通常の入力モードに戻ります。語句や読みの入力の途中でも ESC キーで通常の入力モードに戻ることができます。また、 ESC キーを押さずに引き続き他の語句の登録を行うこともできます。

注）

1. 全候補一覧の表示や番号入力が必要な場合には、単語登録ウィンドウが拡張され、それらも同じウィンドウ上に表示されます。

2. 入力欄に入り切らない未確定文字は表示されませんが、内部的には有効で、変換の対象になります。確定文字が入力欄のサイズを超える場合には余分な部分が削除されます。

3. 確定した文字列の修正の為には、[→]、[←]、[Delete]、[Back Space] キーが使用できます。入力された文字列はカーソルの位置から挿入されます。

1.3 日本語の表示

ここでは、AIX上で使用できる日本語フォントについて説明します。フォントは文字を表示する時の書体を定義するもので、Ja_JPロケールとja_JPロケールの日本語の場合以下の４種類のフォントを同時に使用しています。

ローマン仮名	JISX0201.1976-0
漢字	JISX0208.1983-0
片仮名	JISX0201.1976-1
IBM 拡張文字と NEC 選定文字	IBM-udcJP

使用するフォントを指定するには、X Window System で定義されているX論理フォント名(XLFD)を使います。以下にAIXによって提供されるフォントを示します

- 標準体フォント

ポイント数	書体	ドット数	X論理フォント名
6ポイント	明朝	8	-ibm_aix-mincho-medium-r-normal--8-60-100-100-m-*
9ポイント	ゴシック	12	-ibm_aix-gothic-medium-r-normal--12-90-100-100-m-*
10ポイント	ゴシック	14	-ibm_aix-gothic-medium-r-normal--14-100-100-100-m-*
12ポイント	ゴシック	16	-ibm_aix-gothic-medium-r-normal--16-120-100-100-m-*
12ポイント	ゴシック	18	-ibm_aix-gothic-medium-r-normal--18-120-100-100-m-*
12ポイント	ゴシック	19	-ibm_aix-gothic-medium-r-normal--19-120-100-100-m-*
17ポイント	明朝	26	-ibm_aix-mincho-medium-r-normal--26-170-100-100-m-*
17ポイント	明朝	27	-ibm_aix-mincho-medium-r-normal--27-170-100-100-m-*
23ポイント	明朝	34	-ibm_aix-mincho-medium-r-normal--34-230-100-100-m-*
23ポイント	明朝	35	-ibm_aix-mincho-medium-r-normal--35-230-100-100-m-*
12ポイント	ゴシック	18	-ibm_aix-gothic_new-medium-r-normal--18-120-100-100-m-*
12ポイント	ゴシック	19	-ibm_aix-gothic_new-medium-r-normal--19-120-100-100-m-*
17ポイント	明朝	24	-ibm_aix-mincho-medium-r-normal--24-170-100-100-m-*
23ポイント	ゴシック	34	-ibm_aix-gothic-medium-r-normal--34-230-100-100-m-*
23ポイント	ゴシック	35	-ibm_aix-gothic-medium-r-normal--35-230-100-100-m-*

フォント・エイリアス名				
ポイント数	ローマン仮名	漢字	片仮名	IBM拡張文字とNEC選定文字
6ポイント	RomanKn06	Kanji06	Kana06	IBM_JPN06
9ポイント	RomanKn09	Kanji09	Kana09	IBM_JPN09
10ポイント	RomanKn10	Kanji10	Kana10	IBM_JPN10
12ポイント	RomanKn12C	Kanji12C	Kana12C	IBM_JPN12C
12ポイント	RomanKn12S	Kanji12S	Kana12S	IBM_JPN12S
12ポイント	RomanKn12	Kanji12	Kana12	IBM_JPN12
17ポイント	RomanKn17S	Kanji17S	Kana17S	IBM_JPN17S
17ポイント	RomanKn17	Kanji17	Kana17	IBM_JPN17
23ポイント	RomanKn23S	Kanji23S	Kana23S	IBM_JPN23S
23ポイント	RomanKn23	Kanji23	Kana23	IBM_JPN23
12ポイント	RomanKn12DS	Kanji12DS	Kana12DS	IBM_JPN12DS
12ポイント	RomanKn12D	Kanji12D	Kana12D	IBM_JPN12D
17ポイント	RomanKn17C	Kanji17C	Kana17C	IBM_JPN17C
23ポイント	RomanKn23GS	Kanji23GS	Kana23GS	IBM_JPN23GS
23ポイント	RomanKn23G	Kanji23G	Kana23G	IBM_JPN23G

フォント・ファイル名				
ポイント数	ローマン仮名	漢字	片仮名	IBM拡張文字とNEC選定文字
6ポイント	RomanKn06.pcf.Z	Kanji06.pcf.Z	Kana06.pcf.Z	IBM_JPN06.pcf.Z
9ポイント	RomanKn09.pcf.Z	Kanji09.pcf.Z	Kana09.pcf.Z	IBM_JPN09.pcf.Z
10ポイント	7x14rk.pcf.Z	k14.pcf.Z	7x14rk.pcf.Z	IBM_JPN10.pcf.Z
12ポイント	8x16rk.pcf.Z	jiskan16.pcf.Z	8x16rk.pcf.Z	IBM_JPN12C.pcf.Z
12ポイント	RomanKn12S.pcf.Z	Kanji12S.pcf.Z	Kana12S.pcf.Z	IBM_JPN12S.pcf.Z
12ポイント	RomanKn12.pcf.Z	Kanji12.pcf.Z	Kana12.pcf.Z	IBM_JPN12.pcf.Z
17ポイント	RomanKn17S.pcf.Z	Kanji17S.pcf.Z	Kana17S.pcf.Z	IBM_JPN17S.pcf.Z
17ポイント	RomanKn17.pcf.Z	Kanji17.pcf.Z	Kana17.pcf.Z	IBM_JPN17.pcf.Z
23ポイント	RomanKn23S.pcf.Z	Kanji23S.pcf.Z	Kana23S.pcf.Z	IBM_JPN23S.pcf.Z
23ポイント	RomanKn23.pcf.Z	Kanji23.pcf.Z	Kana23.pcf.Z	IBM_JPN23.pcf.Z
12ポイント	RomanKn12DS.pcf.Z	Kanji12DS.pcf.Z	Kana12DS.pcf.Z	IBM_JPN12DS.pcf.Z
12ポイント	RomanKn12D.pcf.Z	Kanji12D.pcf.Z	Kana12D.pcf.Z	IBM_JPN12D.pcf.Z
17ポイント	12x24rk.pcf.Z	jiskan24.pcf.Z	12x24rk.pcf.Z	IBM_JPN17C.pcf.Z
23ポイント	RomanKn23GS.pcf.Z	Kanji23GS.pcf.Z	Kana23GS.pcf.Z	IBM_JPN23GS.pcf.Z
23ポイント	RomanKn23G.pcf.Z	Kanji23G.pcf.Z	Kana23G.pcf.Z	IBM_JPN23G.pcf.Z

- 太字体フォント

ポイント数	書体	ドット数	X論理フォント名
12ポイント	ゴシック	19	-ibm_aix-gothic-bold-r-normal--19-120-100-100-m-*
17ポイント	明朝	27	-ibm_aix-mincho-bold-r-normal--27-170-100-100-m-*
23ポイント	明朝	35	-ibm_aix-mincho-bold-r-normal--35-230-100-100-m-*
23ポイント	ゴシック	35	-ibm_aix-gothic-bold-r-normal--35-230-100-100-m-*

フォント・エイリアス名				
ポイント数	ローマン仮名	漢字	片仮名	IBM拡張文字とNEC選定文字
12ポイント	RomanKn12B	Kanji12B	Kana12B	IBM_JPN12B
17ポイント	RomanKn17B	Kanji17B	Kana17B	IBM_JPN17B
23ポイント	RomanKn23B	Kanji23B	Kana23B	IBM_JPN23B
23ポイント	RomanKn23GB	Kanji23GB	Kana23GB	IBM_JPN23GB

フォント・ファイル名				
ポイント数	ローマン仮名	漢字	片仮名	IBM拡張文字とNEC選定文字
12ポイント	RomanKn12B.pcf.Z	Kanji12B.pcf.Z	Kana12B.pcf.Z	IBM_JPN12B.pcf.Z
17ポイント	RomanKn17B.pcf.Z	Kanji17B.pcf.Z	Kana17B.pcf.Z	IBM_JPN17B.pcf.Z
23ポイント	RomanKn23B.pcf.Z	Kanji23B.pcf.Z	Kana23B.pcf.Z	IBM_JPN23B.pcf.Z
23ポイント	RomanKn23GB.pcf.Z	Kanji23GB.pcf.Z	Kana23GB.pcf.Z	IBM_JPN23GB.pcf.Z

- 斜字体フォント

ポイント数	書体	ドット数	X論理フォント名
12ポイント	ゴシック	19	-ibm_aix-gothic-medium-o-normal--19-120-100-100-m-*
17ポイント	明朝	27	-ibm_aix-mincho-medium-o-normal--27-170-100-100-m-*
23ポイント	明朝	35	-ibm_aix-mincho-medium-o-normal--35-230-100-100-m-*
23ポイント	ゴシック	35	-ibm_aix-gothic-medium-o-normal--35-230-100-100-m-*

フォント・エイリアス名				
ポイント数	ローマン仮名	漢字	片仮名	IBM拡張文字とNEC選定文字
12ポイント	RomanKn12I	Kanji12I	Kana12I	IBM_JPN12I
17ポイント	RomanKn17I	Kanji17I	Kana17I	IBM_JPN17I
23ポイント	RomanKn23I	Kanji23I	Kana23I	IBM_JPN23I
23ポイント	RomanKn23GI	Kanji23GI	Kana23GI	IBM_JPN23GI

フォント・ファイル名				
ポイント数	ローマン仮名	漢字	片仮名	IBM拡張文字とNEC選定文字
12ポイント	RomanKn12I.pcf.Z	Kanji12I.pcf.Z	Kana12I.pcf.Z	IBM_JPN12I.pcf.Z
17ポイント	RomanKn17I.pcf.Z	Kanji17I.pcf.Z	Kana17I.pcf.Z	IBM_JPN17I.pcf.Z
23ポイント	RomanKn23I.pcf.Z	Kanji23I.pcf.Z	Kana23I.pcf.Z	IBM_JPN23I.pcf.Z
23ポイント	RomanKn23GI.pcf.Z	Kanji23GI.pcf.Z	Kana23GI.pcf.Z	IBM_JPN23GI.pcf.Z

2. ユーザー辞書ユーティリティー

■ 目次

2.1 はじめに

ユーザー辞書ユーティリティーは、ユーザー固有の語句が登録できるユーザー辞書に対して日本語 aixterm 上で管理・保守を行う為のプログラムです。このプログラムは次の機能を提供します。

1. ユーザーがよく使用する語句をユーザー辞書に登録したり、使用しなくなった語句をユーザー辞書から削除します。例えば、

   よく使う人名、地名、専門用語等を登録する。
   よく使う長い文に短い読みを付けて登録する。
   （例：自分の住所を「じゅうしょ」等の読みで登録する。）
   記号に読みを付けて登録する。
   フォント・ユーティリティー*で作成した記号・文字に新しい読みを付けて登録する。
   （*…フォント・ユーティリティーは、「AIX フォント・ユーティリティー　ユーザーズ・ガイド」(SC88-0002)を参照してください。

2. ２つのユーザー辞書を組み合わせて、使用目的に応じたユーザー辞書を作成することができます。

3. ユーザー辞書に登録されている内容を印刷したり、テキスト・ファイルとして出力することができます。

必要なファイルと環境

ユーザー辞書ユーティリティーを実行するには、次のファイルが必要です。

ユーザー辞書ユーティリティー・プログラム
(/usr/bin/kjdict)
ユーザー辞書ユーティリティーのプログラムです。

ユーザー辞書・ファイル
(/usr/lpp/jls/dict/usrdict)
ユーザー辞書・ファイル。ユーザー辞書ユーティリティーを使ってこのファイルの保守・管理ができるのはスーパー・ユーザーだけです。一般ユーザーがユーザー辞書ファイルを保守・管理する時には、このファイルを適当なディレクトリーにコピーして使用してください。

2.2 始動と終了

2.2.1 始動

ユーザー辞書ユーティリティーは、次のコマンドにより始動します。

（形式１）メニュー表示による登録

$ kjdict [-d dictname] [-P printer-queue-name]

例）ユーザー辞書ファイル名が mydict である時

$ kjdict -d mydict

（形式２）コマンド・ライン登録

登録作業のみの場合には次の方法によりコマンド・ラインからの登録もできます。

$ kjdict [-d dictname] [-P printer-queue-name] –y 読み –g 語句

例）読み「どうぶつ」を語句「アフリカ像」で登録したい時

$ kjdict -y どうぶつ –g アフリカ像

（オプションの説明）

-d 登録・変更したいユーザー辞書ファイルを指定します。
-P プリント・キューを指定します。
-y 読みを指定します。
-g 語句を指定します。

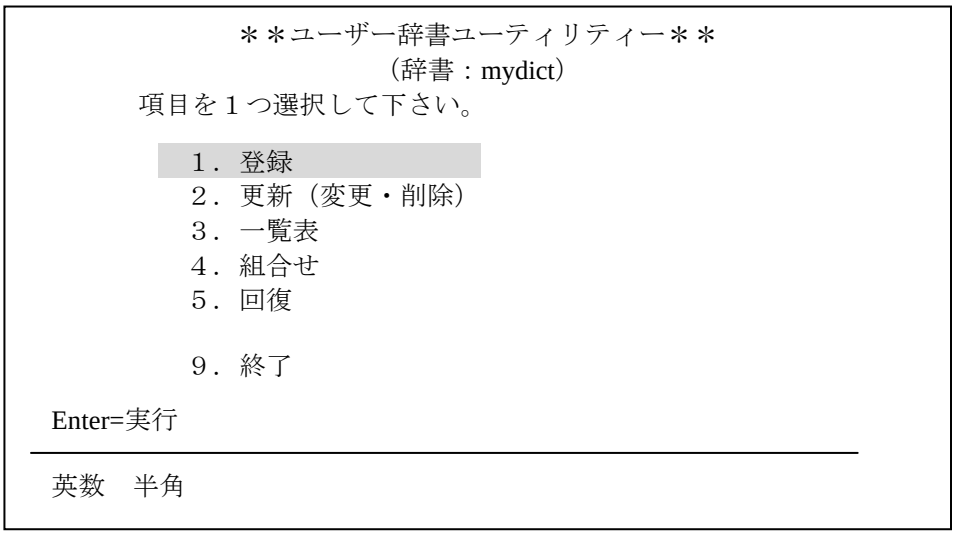
ユーザー辞書ファイル名を指定しなかった場合、次の順序でユーザー辞書を参照します。

1. 環境変数で設定されているユーザー辞書
   環境変数で JIMUSRDICT＝としてファイル名が指定されていれば、そのファイルを参照します。

2. ホーム・ディレクトリーのユーザー辞書。($HOME/.usrdict)

3. 決まったパス名のユーザー辞書。(/usr/lpp/jls/dict/usrdict)

コマンド kjdict を日本語 aixterm 上で実行すると、次の初期メニューが表示されます。



それぞれのメニューの項目は、次の２通りの方法で選択することができます。

番号で選択する

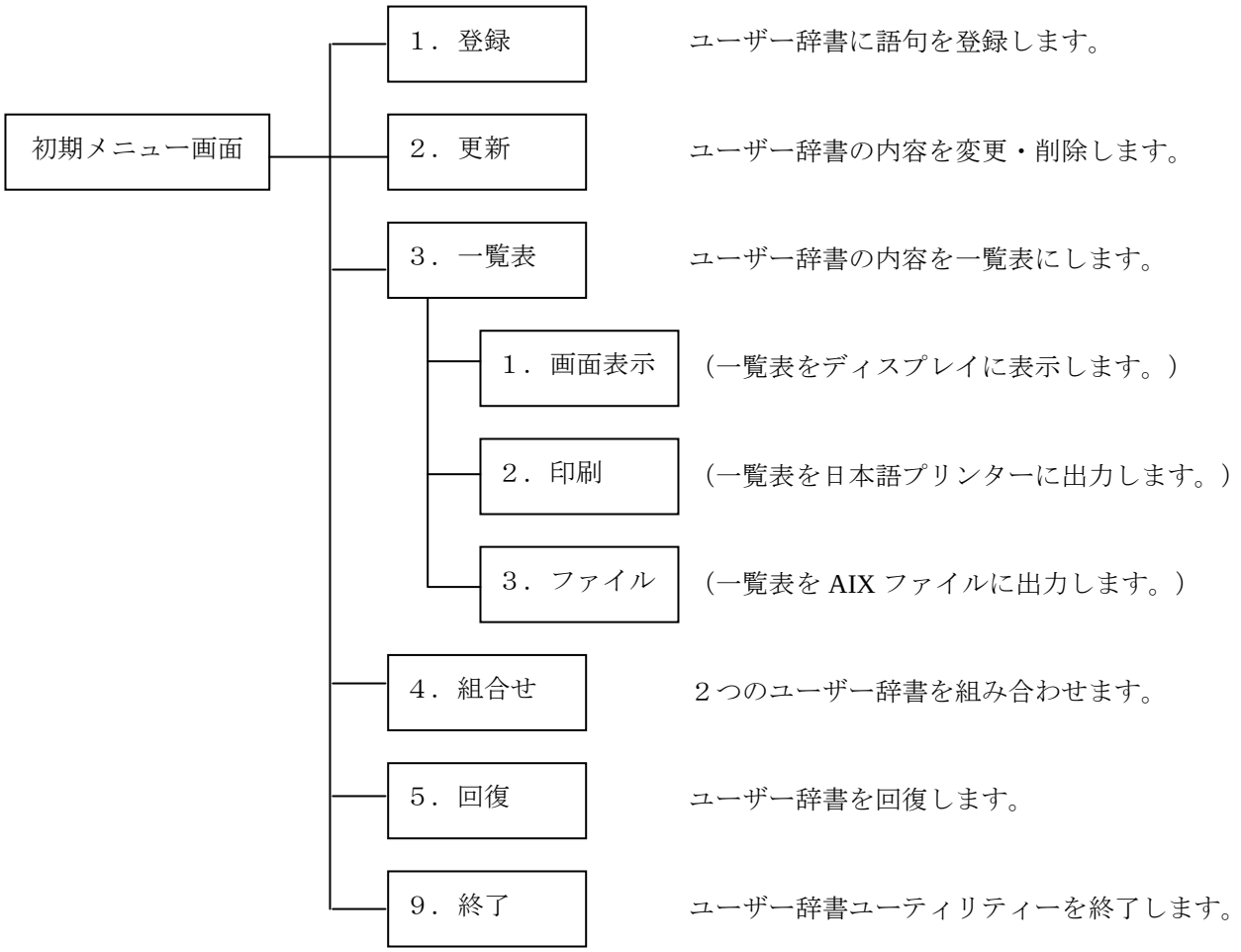
作業を行いたい項目の番号（半角文字）を入力すると、その項目が選択されます。

カーソル移動キー（上下）を用いて選択する

1. 選択する項目にカーソルを移動してください。移動先の項目が反転されます。
2. Enter キーを押してください。その項目が選択されます。

タイトル「ユーザー辞書ユーティリティー」の下には、登録・変更しようとしているユーザー辞書ファイル名が表示されます。（上図の例では mydict）

作業中は一時的に作業用ファイルとして、”ユーザー辞書ファイル名.tmp”という名前ファイルがカレント・ディレクトリーに作られます。



2.2.2 終了

初期メニュー画面から９．終了を選択してください。ユーザー辞書ユーティリティーは終了します。

ユーザー辞書ユーティリティーを開始する前の、ユーザー辞書ファイルは”ユーザー辞書ファイル名.bak”の名前で、ユーザー辞書ファイルと同じディレクトリーに保管されます。

操作上の留意点

ユーザー辞書ユーティリティーの終了には、必ず初期メニュー画面で９．終了を選択するようにしてください。それ以外の方法で終了した場合（kill コマンドによる強制終了等）、登録・更新した内容は反映されません。

ユーザー辞書ユーティリティーの終了後、新たに aixterm を始動させます。その aixterm 上で新しいユーザー辞書が使用できます。

２つ以上のユーティリティーを、同じユーザー辞書に対して同時に始動した場合、ユーザー辞書ファイルは正しく保存されないことがあります。

ユーザー辞書のファイル名は、1～10 カラム長（半角 10 文字／全角 5 文字）にしてください。

ユーザー辞書ユーティリティーは、dtterm 上では使用できません。

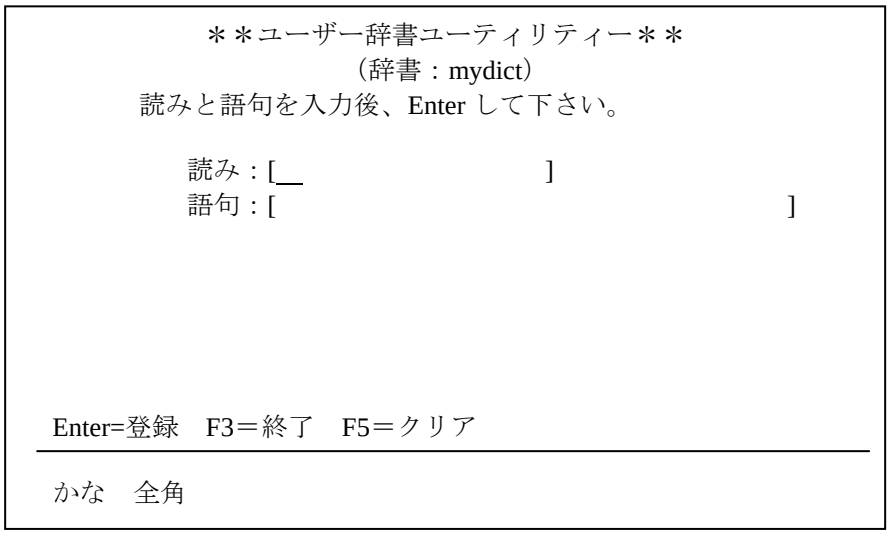
2.3　ユーザー辞書への登録

2.3.1 機能

システム辞書にない読みを登録したり、ユーザーがよく使用する語句に読みを付けて登録します。

2.3.2 操作手順

1. 初期メニュー画面から１．登録を選択してください。次の登録画面が表示されます。



2. この画面において、読みと語句をそれぞれの入力フィールドに入力します。

カーソルを一方の入力フィールドからもう一方の入力フィールドへ移動させるには、↑（上矢印）キー、↓（下矢印）キー、または ⇔（タブ・バックタブ）キーを使います。

読みと語句にはそれぞれ以下のような制限があります（いずれも全角文字)。

	読み	語句
使用できる文字	ひらがな カタカナ（但し、登録時にひらがなに変換される。） 英大文字 英小文字（但し、登録時に英大文字に変換される。） 数字 記号　！＃％＆()＊+－／：；＜＝＞？@	$ 以外の文字
文字の混在	ひらがなとカタカナの混在、英文字と数字の混在、 または記号どうしの混在のみ許される。	制限なし
文字数	ひらがな・カタカナの場合は 10 文字以内。英数・ 記号の場合は 9 文字以内。	20 文字以内

同一の読みに対して登録できる語句の字数（データ量）には、一定の制限があります。以下を参考にしてください。
同一の読みに対して、5 文字の語句を 23 個まで登録できます。
同一の読みに対して、10 文字の語句を 12 個まで登録できます。

3. 読みと語句を入力した後 Enter キーを押します。

   『登録されました』というメッセージが表示されれば、入力した読みと語句がユーザー辞書に登録されたことを示します。

   その他、読みや語句に不備な点があった時や辞書の制限を超える場合は、それを知らせるメッセージが表示されます。

4. F5 キーにより、入力フィールドおよびメッセージを消去することができます。

5. 登録作業を終えたら F3 キーを押してください。初期メニュー画面に戻ります。

操作上の留意点

読みに対する語句は一度に 1 つしか登録できません。 1 つの読みに複数の語句を登録する場合、 1 つ登録するごとに Enter キーを押して下さい。

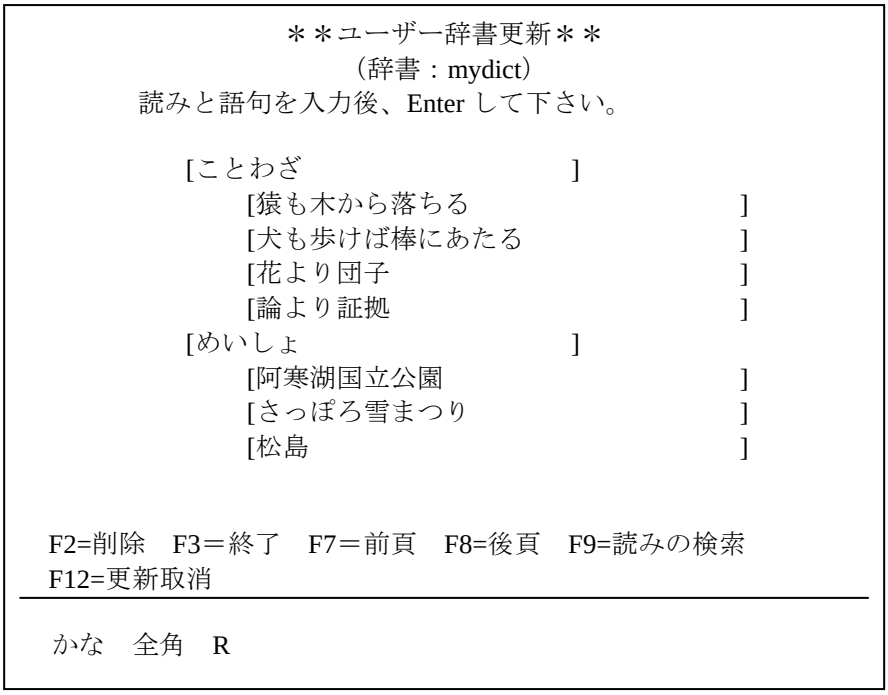
2.4　ユーザー辞書の更新

2.4.1 機能

ユーザー辞書に登録されている読みと語句を変更・削除します。

2.4.2 操作手順

1. 初期メニュー画面の２．更新（変更・削除）を選択してください。次に更新画面が表示されます。



ユーザー辞書に登録されている読みは、次の順序で表示されます。
数字（昇順）→ 英字（ABC 順）→ ひらがな（あいうえお順）

2. この更新画面においてユーザー辞書の内容を変更・削除します。
カーソルを任意の位置に移動して、読み、語句を更新してください。カーソルの移動 ↑ （上矢印）キー、↓ （下矢印）キー、また ⇔ （タブ・バックタブ）キーを使います。
また F7 キーで前のページに戻り F8 キーで次のページに進みます。

１つの読みに対する語句が複数であり、それが前のページまたは次のページにまたがっている場合、そのことを知らせる為に↑、↓というサインが表示されます。

```
                ＊＊ユーザー辞書更新＊＊
                    （辞書：mydict）
          [ことわざ                    ]
            ↑ [論より証拠                    ]
              [鶴は千年、亀は万年              ]
              [転ばぬ先の杖                    ]
              [七転び八起き                    ]
          [めいしょ                    ]
              [阿寒湖国立公園                  ]
              [さっぽろ雪まつり                ]
            ↓ [松島                            ]


  F2=削除  F3＝終了  F7＝前頁  F8=後頁  F9=読みの検索
  F12=更新取消
 ------------------------------------------------------------
  かな  全角  R
```

更新作業の詳しい操作方法は、この後のそれぞれの項を参照して下さい。

3. 更新作業を全て終えたら F3 キーを押して下さい。初期メニュー画面に戻ります。
作業に誤りがあって更新作業を取り消す時は、 F12 キーを押します。次のメッセージが表示されます。

```
更新を取り消します。Enter=更新しない  F12＝更新機能
```

ここで Enter キーを押せばユーザー辞書は更新されずに初期メニュー画面に戻ります。
更新を続けるには F12 キーを押します。

2.4.3 登録されている読み・語句を削除する

登録されている読み・語句をユーザー辞書から削除します。
読みを削除すると、その読みで登録されている語句も全て削除されます。
また、登録されている語句を全て削除すると対応する読みも削除されます。

1. 削除する読みまたは語句のフィールドへカーソルを移動させてください。

2. F2 キーを押して下さい。削除の対象となる読み、語句が反転表示され、削除の確認メッセージが表示されます。

例）
語句『ニホンザル』のところへカーソルを移動させ、 F2 キーを押した場合。

```
      [どうぶつ                    ]
            [アフリカ象                ]
            [ニホンザル                ]

  ユーザー辞書から削除します。Enter=削除  F12＝取消
```

3. Enter キーを押すことにより、反転表示部分が削除されます。削除を取り消す場合は F12 キーを押します。

2.4.4 登録されている読みを探す

変更、または削除したい読み・語句が最初から画面に表示されているとは限りません。ユーザー辞書に登録されている読み・語句が多い場合、「読みのサーチ機能」を使うことにより目的の読みをすばやく表示させることができます。

1. F9 キーを押して下さい。次のようなメッセージが表示されます。

```
読みを入力後 Enter キー[_                ]
```

2. メッセージの後に目的の読みを入力します。

3. Enter キーを押します。

4. 入力された読みと一致する読みがあれば、その読みが画面の最上行となるように表示します。
   一致するものがなければ、入力された読みに最も近い読みが選ばれます。

5. メッセージが表示されている状態で読みを探すことを中止するには、何も入力せず Enter キーを押してください。

操作上の留意点

読みと語句の入力には一定の規則がありますので、制限内の変更を行います。
文字の制限については『ユーザー辞書に語句を登録する』の項を参照してください。

読みを、すでにユーザー辞書にある読みには変更できません。

変更した読みや最初に不備な点があった時には、その読みまたは語句が反転表示され、エラー・メッセージが表示されます。その場合 F12 キーを押すことにより、元の読みまたは語句に戻ります。

システム辞書と同一の読み・語句の組合せは登録できません。読みまたは語句を変更した結果、その組合せがすでにシステム辞書に存在する場合、その語句は反転表示され、その語句を削除しても良いかを確認するメッセージが表示されます。

例）

```
[ふるーつ              ]
        [葡萄                  ]
        [果物                  ]

          ↓   『ふるーつ』を『くだもの』に変更

[くだもの              ]
        [葡萄                  ]
        [果物                  ]

  この読みと語句はすでにシステム辞書に登録されています。
  この語句を削除していいですか？    Enter=削除   F12=取消
```

削除して良ければ Enter キーを押します。残したければ F12 キーを押します。F12 キーを押した場合、読みは変更される前の読みに戻ります。

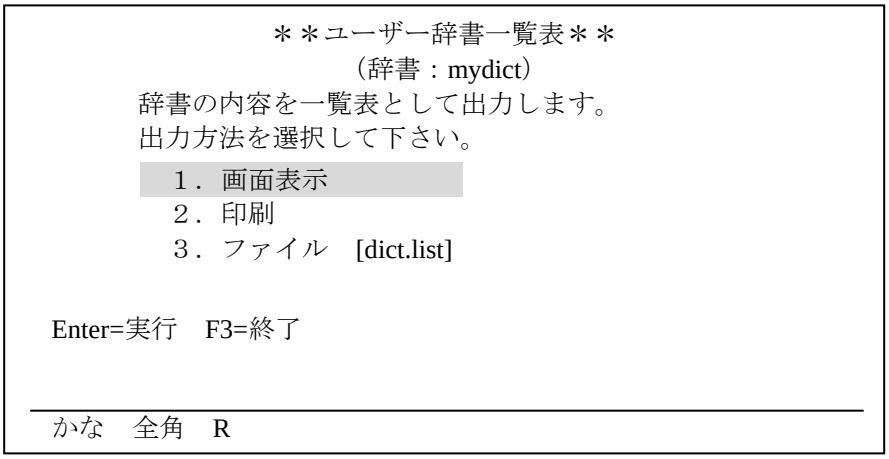
2.5　ユーザー辞書の一覧表作成

2.5.1　機能

ユーザー辞書に登録されている内容を一覧表にして出力します。

2.5.2　操作手順

1. 初期メニュー画面の３．一覧表を選択してください。次の一覧表画面が表示されます。

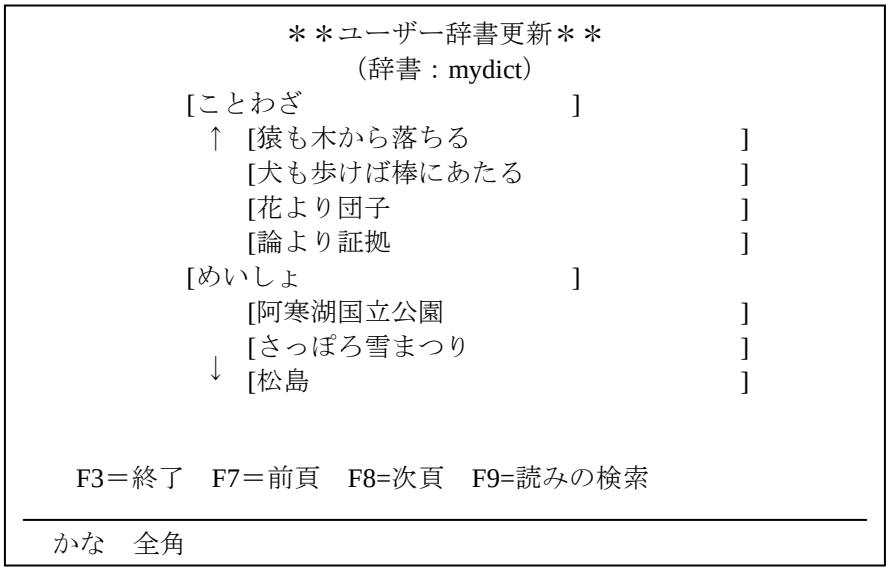


2. 一覧表の出力方法は、ディスプレイ、プリンター、ファイルの 3 種類あります。
任意の方法を画面から選んでください。
（その選択方法は、前述した初期メニュー画面での選択方法と同様です。）

2.5.3　画面表示（ディスプレイに表示する）

ユーザー辞書に登録されている内容を一覧表として画面に表示します。

1. 一覧表画面の１．画面表示を選択してください。次の画面が表示されます



2. ファンクション・キー（ F7 , F8 , F9 ）により目的の読みを表示します。

3. 一覧表示に戻るには、 F3 キーを押します。

注）この画面ではユーザー辞書の更新はできません。ユーザー辞書の更新は更新画面でないとできません。
（『ユーザー辞書の内容を変更・削除する』を参照してください。）

2.5.4 印刷

ユーザー辞書に登録されている内容を一覧表として日本語プリンターに出力します。

1. 一覧表画面の２．印刷を選択してください。

2. 印刷要求がプリンターに送られると（待ち行列に入ると）、次のメッセージが表示されます。

```
一覧表の印刷は終了しました。
```

注）ユーザー辞書の一覧表は、プリント・キューの指定が無い場合には、/etc/qconfig ファイルに記述されている最初のデバイス（通常は /dev/lp0）に出力されます。kjdict 始動時にプリント・キューの指定をすることができます。

例）ユーザー辞書ファイル mydict をプリント・キューrp0 で指定する時

```
$ kjdict –d mydict –P rp0
```

2.5.5 ファイル

ユーザー辞書に登録されている内容を一覧表としてテキスト・ファイルに出力します。

1. 一覧表画面の３．ファイルを選択して下さい。次の位置にカーソルが表示されます。

```
３．ファイル　[dict.list]
```

2. 任意のファイル名を入力して下さい。尚、最初は dict.list というファイル名が省略ファイル名として表示されています。

3. ファイル名を入力し終えたら、 Enter キーを押します。正常にファイルができた場合には、次のメッセージが表示されます。

```
一覧表のファイル出力は終了しました。
```

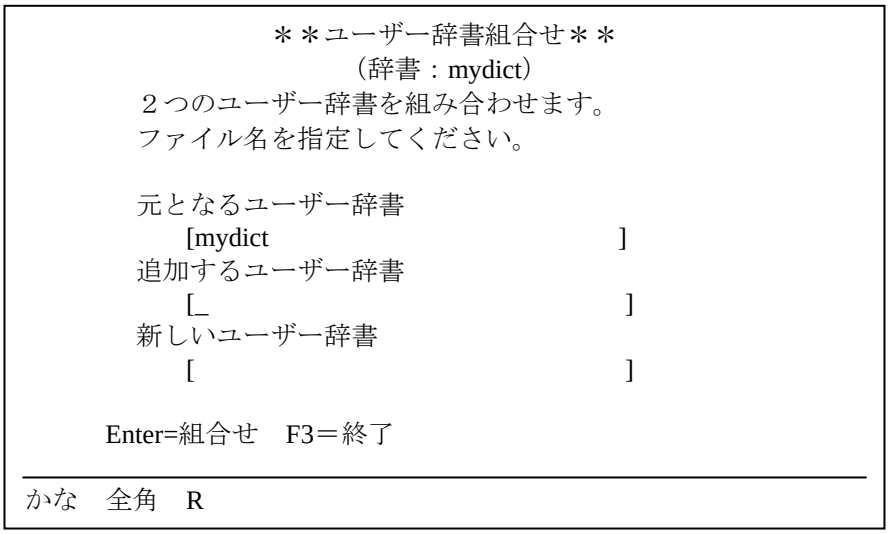
2.6　ユーザー辞書の組合せ

2.6.1　機能

現在作業中のユーザー辞書に他のユーザー辞書を組合せ、新しいユーザー辞書を作成します。

2.6.2　操作手順

1. 初期メニュー画面の４．組合せを選択して下さい。次の組合せ画面が表示されます。



2. 『追加するユーザー辞書』および『新しいユーザー辞書』のファイル名をそれぞれ入力します。

3. Enter キーを押すと、ユーザー辞書の組合せが行われます。

   『ユーザー辞書の組合せを実行中です。』というメッセージに引き続き、『ユーザー辞書の組合せが終了しました。』というメッセージが表示されれば、ユーザー辞書の組合せが正常に終了したことを示します。

2.7　ユーザー辞書の回復

2.7.1 機能

ユーザー辞書の回復を行います。
『ユーザー辞書の回復が必要です。』というメッセージが表示された時に、この機能を作動させます。

2.7.2 操作手順

初期メニュー画面の５．回復を選択してください。

処理中は、

```
  ユーザー辞書を回復中です。
```

というメッセージが表示されます。
処理が正常に終了すると、

```
  ユーザー辞書の回復が終了しました。
```

というメッセージが表示されます。

注）このとき、不正な語句がユーザー辞書から削除されることがあります。

2.8　ユーザー辞書の使用方法

新たに登録・更新されたユーザ辞書ファイルを端末エミュレーター等のアプリケーション・プログラムで使用するには、次のようにカストマイズしてください。

例１）ユーザー辞書ファイルのパス名が /u/user/dict/myidct の場合、次のように環境変数 JIMUSRDICT を設定してください。

```
JIMUSRDICT=/u/user/dict/mydict
export JIMUSRDICT
```

例２）ユーザー辞書ファイルを、ファイル名.usrdict としてホームディレクトリーの下へコピーしてください。

注）この時、環境変数 JIMUSRDICT は設定しないでください。

ユーザー辞書ファイルを正しく設定した後、アプリケーション・プログラムを始動してください。
そのアプリケーション・プログラム上で新しいユーザー辞書が使用できます。

2.9 メッセージ

以下に、いくつかのメッセージに対する説明と処置についてまとめました。

2.9.1 始動に関連したメッセージ

ターミナル ‘ XXXXX ’ 上では、このプログラムは実行できません。
(Invalid terminal type ‘ XXXXX ’.)

説明：無効な表示装置 ‘ XXXXX ’ 上で、kjdict を実行しようとしました。
処置：kjdict はグラフィック表示装置上で、X ウィンドウ日本語ターミナルエミュレーター(aixterm)から実行します。

グラフィック表示装置で実行できない場合は、TERM 環境変数をチェックしてください。

```
TERM=aixterm
export TERM
```

等の設定がされていない場合は、適切な環境変数で設定してください。

無効なオプション ‘ XXXXX ’ が指定されました。

説明：コマンド入力で実行した時に指定したオプションが正しくありません。
処置：ユーザー辞書ファイルの指定に使うオプションは ’-d’ なので、

```
$kjdict –d ユーザー辞書ファイル名
```

の形で指定します。’-d’ の直後にはブランクが必要です。

ユーザー辞書 ‘ XXXXX ’ がありません。

説明：指定されたユーザー辞書ファイルは存在せず、オープンすることができませんでした。
処置：ユーザー辞書ファイル名を正しく指定します。

ユーザー辞書 ‘ XXXXX ’ の書き込み（読み取り）が許可されません。

説明：指定されたファイルはそのユーザーに対して許可されていないので、オープンすることができませんでした。
処置：ユーザー辞書については、読み取りと書き込みが許可されているファイルを指定します。

ディレクトリー ‘ XXXXX ’ の書き込みが許可されません。

説明：１）ユーザー辞書ファイルが保管されているディレクトリーに対し許可されていないので、ユーザー辞書ファイルの変更ができません。
２）カレント・ディレクトリーに対し書き込みが許可されていないので、作業用ファイルが作成できません。

処置：１）書き込みが許可されているディレクトリーのユーザー辞書ファイルを指定します。
２）ユーザー辞書ユーティリティー実行中は、カレント・ディレクトリーに作業用ファイルが作成されます。従って、書き込みが許可されているディレクトリーの下でユーザー辞書ユーティリティーを実行する必要があります。

画面サイズが小さすぎてプログラムが実行できません。

説明：画面サイズがとても小さい日本語 aixterm で、kjdict を実行しようとしました。
処置：aixterm の画面サイズを以下の数値よりも大きくします。
　　横 54 カラム　横 17 カラム
　　（半角文字の場合．．．54×17、全角文字の場合．．．27×17）

ユーザー辞書にアクセス・エラー、終了してください。

説明：ユーザー辞書ファイルに対し、アクセス・エラーが起こりました。
処置：初期メニュー画面で「終了」を選択してユーティリティーを終了してください。ユーザー辞書ファイルを点検後、再度実行を試みてください。

システム辞書にアクセス・エラー、終了してください。

説明：システム辞書ファイルに対し、アクセス・エラーが起こりました。
処置：初期メニュー画面で「終了」を選択してユーティリティーを終了してください。システム辞書ファイルを点検後、再度実行を試みてください。

システム・エラーが起こりました。終了してください。

説明：システム上に何らかのエラーが起こり、ユーティリティーを実行することができません。
処置：初期メニュー画面で「終了」を選択してユーティリティーを終了してください。システムを点検して原因を調べてください。

ユーザー辞書はすでに更新中です。処理を強制的に開始しますか？　Enter=開始　F12=取消

説明：そのユーザー辞書に対し、他のユーティリティーが更新中か、前回の更新作業が途中で中断された為に、辞書ファイルが更新中の状態になっています。
処置：他のユーザーが更新作業中であるならば F12 キー を押して終了するのを待ちます。誰もそのユーザー辞書ファイルに対し更新していなければ Enter キーを押してユーティリティーを開始します。

　　注）同じユーザー辞書ファイルに対し、ユーティリティーを複数始動しないでください。ユーティリティーを複数始動した場合、ユーザー辞書ファイルは正しく保存されていないことがあります。

ユーザー辞書にデータがありません。

説明：現在作業中のユーザー辞書にデータがない時に、初期メニュー画面において更新／一覧表／組合せのいずれかの項目が選択されました。
処置：更新／一覧表／組合せの作業をする時は、ユーザー辞書に何らかのデータが登録されている必要があります。初期メニュー画面で『１．登録』を選択し、ユーザー辞書にデータを登録してから更新／一覧表／組合せの作業を行ってください。

2.9.2 登録／更新に関連したメッセージ

読みと語句が同じです。登録できません。

説明：読みと語句の入力フィールドに、全く同一の文字列を入力して登録しようとしました。
処置：読みと語句が同じものは登録する必要がありません。

読みとして無効な文字が含まれています。

説明：読みの入力フィールドに、読みとして無効な文字が入力されています。
処置：読みとして有効な文字だけになるように修正します。

ひらがなと英数字の混在は読みとして無効です。

説明：ひらがなと英数字・記号の混じった文字列を読みとして入力して登録しようとしました。
処置：読みをひらがなのみの文字列、または英数字・記号のみの文字列となるように修正します。

英数文字の最大長（9 文字）を超えています。

説明：英数字・記号で 10 文字以上からなる文字列を読みとして登録しようとしました。
処置：英数字・記号の読みの場合、9 文字までです。

この読みと語句はすでにユーザー辞書に登録されています。

説明：登録しようとした読みと語句はすでにユーザー辞書に登録されています。
処置：＜登録作業中の場合＞登録する必要はありません。
＜更新作業中の場合＞更新できないので、必要がなければ削除します。

この読みと語句はすでにシステム辞書に登録されています。

説明：登録しようとした読みと語句はすでにシステム辞書に登録されているものと同一です。
処置：＜登録作業中の場合＞登録する必要はありません。
＜更新作業中の場合＞更新できないので、必要がなければ削除します。

この読みにはこれ以上語句を登録できません。

説明：同一の読みに対して登録できる語句の字数（データ量）には制限があり、その範囲を超えてしまうので登録できません。
処置：その語句がどうしても必要であれば、違う読みで登録します。

ユーザー辞書がいっぱいです。

説明：ユーザー辞書ファイルにデータを書き込む余地がなくなります。
処置：＜登録作業中の場合＞登録作業を中止します。
＜更新作業中の場合＞更新画面で不要な語句を削除します。

読みと語句が同じです。変更できません。

説明：読みと語句が全く同一の文字列となるように変更しようしました。
処置：読みと語句が同じものは不要なので、他のものに変更する必要がなければ削除します。

この読みに対する語句の総字数が制限範囲を超えました。

説明：同一の読みに対して登録できる語句の字数（データ量）には制限があり、その範囲を超えてしまうので変更できません。
処置：その読みに対し不要な語句があればそれを削除します。

すでにユーザー辞書にある読みには変更できません。

説明：すでに同一の読みがユーザー辞書にある場合、それと同じ読みに変更することはできません。
処置：更新作業を終了させて、初期メニュー画面で「登録」を選び、読みと語句を登録します。

ユーザー辞書にデータがなくなりました。
F3 キーまたは F2 キーを押してください。

説明：更新作業によりユーザー辞書のデータが全て削除され、更新作業を続けることができません。
処置： F3 キー（更新終了）または F12 キー（更新取消）を押して、更新作業を終了してください。

2.9.3 一覧表に関したメッセージ

プリンター出力でエラーが発生しました。

説明：印刷処理実行中に、プリンターでエラーが発生しました。
処置：プリンター関連の環境をチェックしてください。

2.9.4 組合せに関連したメッセージ

指定のファイルはオープンできません

説明：組合せの実行に必要なファイルがオープンできません。
処置：他のファイルを指定するか、一旦処理を終了してファイルの許可モードを変更してください。

追加するユーザー辞書が存在しません。

説明：組合せに使用されるユーザー辞書が存在しません。
処置：他のユーザー辞書を指定してください。

同じファイル名を指定することはできません。

説明：指定された『元となるユーザー辞書』と『追加するユーザー辞書』が同じです。
処置：他のユーザー辞書を『追加するユーザー辞書』として指定してください。

指定の辞書はユーザー辞書ではありません。

説明：組合せ処理時に指定された『追加するユーザー辞書』が、ユーザー辞書ではありません。
処置：他のユーザー辞書を『追加するユーザー辞書』として指定してください。

指定の辞書は回復が必要です。

説明：組合せ処理時に指定された『追加するユーザー辞書』が、回復処理を必要とします。
処置：他のユーザー辞書を『追加するユーザー辞書』として指定するか、回復処理を実行してください。

指定の辞書は現在使用中です。

説明：組合せ処理時に指定された『新しいユーザー辞書』を、現在他のユーザーが使用中です。
処置：他のユーザー辞書を『新しいユーザー辞書』として指定してください。

ファイルは変更されます。Enter=実行　F12=取消

説明：組合せ処理時に指定された『新しいユーザー辞書』は、すでに存在しています。
処置：変更したくない場合は F12 キーをおして『新しいユーザー辞書』を指定し直してください。

組合せは終了しましたが、一部データが入りませんでした。

説明：組合せを実行中に同一の読みに対する語句の総字数（登録可能なデータ量）が超えてしまったが、ユーザー辞書ファイルにデータを書き込む余地がなくなってしまいました。その為、『新しいユーザー辞書』には一部のデータが入りませんでした。
処置：『元となるユーザー辞書』と『追加するユーザー辞書』の中の、不必要な読み・語句を削除し、再び組合せを実行します。

2.9.5 回復に関連したメッセージ

ユーザー辞書の回復が必要です。

説明：何らかの理由でユーザー辞書が使用不可能であり、回復させる必要があります。
処置：初期メニュー画面で「回復」を選択します。

ユーザー辞書の回復は必要ありません。

説明：ユーザー辞書の回復の必要がないのに、初期メニュー画面で「回復」が選択されました。
処置：必要ありません。

ユーザー辞書の回復が終了しました。

説明：ユーザー辞書のデータが回復し、使用可能となりました。
処置：必要ありません。

回復が終了しましたが、データの一部が失われました。

説明：ユーザー辞書は使用可能となりましたが、登録されていたデータの一部が失われました。
処置：必要ありません。

回復が終了しましたが、データの全てが失われました。

説明：ユーザー辞書は使用可能となりましたが、登録されていたデータの全てが失われました。
処置：必要ありません。

回復ができませんでした。終了してください。

説明：ユーザー辞書の回復を図りましたが、できませんでした。
処置：初期メニュー画面で「終了」を選択します。ユーティリティーを終了させた後、バックアップ・ファイルがあればそれをコピーして使用します。もしなければ、AIX を再導入して再実行してください。

2.10 mksdict コマンド

mksdict コマンドは、AIX日本語入力機能で使用するシステム辞書を作成するためのツールです。AIXが提供するシステム辞書以外に一般的に使用する辞書が必要な場合は、このコマンドを使って作成できます。

mksdict コマンドの使用方法を以下に説明します。

機能：

日本語入力（かな漢字変換）で使用するシステム辞書を作成します。

書式：

```
mksdict data_file new_sysdict
```

機能説明：

mksdict コマンドは、日本語入力（かな漢字変換）で使用するシステム辞書を作成します。読みとそれに対応する漢字を一行に一組ずつ記述したデータファイルを読み込み、指定された名前でシステム辞書を新たに作成します。読みおよび漢字は、データファイル中にSJIS文字列で記述し、それらは１つ以上の空白（１バイト）で区切る必要があります。登録した語句はシステム辞書中で名詞として扱われます。

パラメータ：

data_file	読み込むデータファイルを指定します。
new_sysdict	新たに作成するシステム辞書名を指定します。

使用例：

次のような内容のname_dataファイルがあるとします。

```
いとう          伊藤博文
なつめ          夏目漱石
にとべ          新渡戸稲造
もり            森鴎外
さいごう        西郷隆盛
さかもと        坂本龍馬
おおくま        大隈重信
つぼうち        坪内逍遥
おおくぼ        大久保利通
まさおか        正岡子規
```

以下のコマンドを実行するとname_dictという名前で新しいシステム辞書が作成されます。

```
mksdict name_data name_dict
```

JIMMULDICT環境変数に追加することで、この新しいシステム辞書をかな漢字変換で使用することができます。（以下の例では新しいシステム辞書はディレクトリ/u/myhome/dict/の下にあるものとしています。）

```
JIMMULDICT=/usr/lpp/jls/dict/ibmbase:/usr/lpp/jls/dict/ibmcnnc:
          /usr/lpp/jls/dict/ibmzipc:/u/myhome/dict/name_dict
export JIMMULDICT
aixterm -lang Ja_JP
```

この新しく起動されたaixtermではname_dictを使ってかな漢字変換ができます。

注) mksdict は/tmpディレクトリに作業用のファイルを一時的に作成します。/tmpディレクトリに十分な空き領域が無い場合にはシステム辞書が正しく作成されない場合があります。/usr/lpp/jls/dict/ibmbase はAIXが提供する基本システム辞書です。この辞書は常にJIMMULDICT環境変数の先頭に指定して下さい。

3. 日本語プリンター・サポート

■ 目次

3.1 はじめに

この章では、日本語プリンターの接続方法、オペレーティング・システムへの登録方法、印刷時のオプション等について説明します。

3.1.1 プリンターのインストール

各種プリンターを接続するには、必要なプリンター・サポートのファイルセットをインストールしてください。

印刷プロセス、プリンター構成についての詳細な情報は、以下のマニュアルを参照してください。

「AIX バージョン6.1 プリンターおよび印刷」(SC88-4592)

3.1.2 使用可能なデータ・ストリーム

各プリンターにおいて以下のデータ・ストリームが扱えます。

プリンターの種類	日本語データ・ストリーム
Canon LASER SHOT	LIPS2+, LIPS3, LIPS4
ESC/P J84 printer	ESC/P
IBM PAGES printer	PAGES
OKI MICROLINE	PostScript

データ・ストリームの詳細技術情報は、各プリンターのマニュアルを参照してください。

3.1.3 AIX 側のインターフェース設定

IBM の各種プリンターは 2 種類のハードウェア・インターフェースを持っており、日本語モード時及び英語モード時で異なる場合があります。これらのプリンターを使用する為には、プリンターのインターフェースと AIX のインターフェースを以下のように一致させる必要があります。AIX のインターフェースは、SMIT、mkdev、chdev 等のコマンドで設定及び変更ができます。

通常、IBM 製日本語プリンターはプリンターインターフェースを converged に設定します。それ以外のプリンターはプリンターインターフェースを standard に設定します。

Canon LASER SHOT は standard に設定してください。

注）PCI モデルの一部には converged インターフェースをサポートしていないモデルがあります。

3.1.4 仮想プリンターの定義方法

プリント･ジョブは一般的に仮想プリンターに送られます（qprt、lp あるいは lpr コマンドを使用した場合）。仮想プリンターとは論理的な装置で、プリンター 1 台に対し複数台の設定ができます。特にプリンターが何種類かのデータ・ストリームをサポートしている場合、各データ・ストリームに対して 1 台の仮想プリンターを割り当てることにより、各データ・ストリーム専用のプリンターが存在しているかのように扱うことができます。
仮想プリンターを定義するには以下の 2 つの手順が必要です。

1. プリンター・デバイスの定義
   SMIT を使用して/dev/lpX（一般的に/dev/lp0）というプリンター用デバイス・ファイルを定義します。この時 3.1.3 で述べたインターフェースを指定します。

2. プリント・キュー／プリント・キュー・デバイスの定義と仮想プリンターの設定
   仮想プリンターを実現している機能の 1 つにプリンター・フォーマッターと呼ばれるフィルターがあり、これが印刷出力のフォーマットを決定したり、プリンターをコントロールしたりします。仮想プリンターの設定とは、簡単に言えばキュー及びキューデバイスとプリンター・フォーマッターを関連付けることです。

SMIT で印刷キューを定義することにより、キュー／キュー・デバイスと仮想プリンターを同時に設定します。

SMIT の使用手順は次の通りです。

① SMIT を起動します。

② ーデバイス
　ープリンタ／プロッタ
　　ープリンタ／プロッタ・デバイス
　　　ープリンタ／プロッタの追加
　を選択します。

③ タイプ、インターフェース、親アダプタを選択し、各種パラメータを入力してプリンター・デバイを作成します。

④ プリンタ／プロッタの画面まで戻ります。

⑤ ー印刷スプーリング
　ー印刷キューの追加
　を選択します。

⑥ 接続タイプ、プリンタ・タイプ、プリンタを選択し、印刷キュー名を入力して仮想プリンターを作成します。

3.2 qprt コマンド

qprt コマンドは指定されたファイルを出力する為のプリント・ジョブをキューに登録する為に使うことができるコマンドです。qprt コマンドでは以下のフラグが使用できます。使用可能なフラグまた各フラグに対するオプションのとりうる値が異なるものがあります。

3.2.1 フラグの説明

-P <queue> [:<Queue Device>]

<Queue> プリント・キューの名前
<Queue Device> プリント・キュー・デバイスの名前

このフラグが指定されていない時は、環境変数 PRINTER で指定されたプリント・キューの名前が使われます。環境変数が指定されていない時はシステム・デフォルト・キューが使われます。システム・デフォルト・キューとは、システム・ファイルである/etc/qconfig の中で最初に名前が登録されているキューを意味します。

-d <Input Data type>

<Input Data type> 出力するファイルのデータ・ストリームの種類
<Input Data type> は以下の内いずれか 1 つを指定します。

a 普通のテキスト・ファイルを出力する時に指定します。
入力されたファイルはプリンター・フォーマッターで処理されてから、プリンターに出力されます。
OKI MICROLINE、IBM Network Printer 以外はこれがデフォルトです。

p アプリケーション・プログラムで各プリンターのデータ・ストリームを生成している場合等に使用します。入力されたファイルはプリンター・フォーマッターを通らずにそのままプリンターに出力されます。

s ポストスクリプト用のデータ・タイプです。OKI MICROLINE、IBM Network Printer のみ指定可能です。

-X <Code Page Name>

<Code Page Name> プリントしたいファイルのコード・ページ
<Code Page Name> は以下の内いずれか 1 つを指定します。

IBM-943 デフォルトのコード・ページです。入力されるファイルは PC コード(Ja_JP)として処理されます。Canon LASER SHOT ではプリンターの設定メニューで優先漢字を JIS78 にしておく必要があります。

IBM-932 AIX4.3.1 以前でのデフォルトのコード・ページです。入力されるファイルは PC コード(Ja_JP)として処理されます。

IBM-eucJP 入力されるファイルは EUC コード(ja_JP)として処理されます。

ページ・フォーマットに関するフラグ

-l <Num>
  <Num>　　1 ページあたりの行数
　　0 の場合はページ長を無視して、ひとつの継続したページとみなされます。

ただし、ESC/P J84 Printer では以下の属性値でもページ長が設定されています。
　　WU　　最大ページ長 (インチ 単位)
Canon LASER SHOT では以下の属性値でもページ長が設定されています。
　　wI　　下部マージン (mV 単位)
したがって、ページ長を無視するにはこれらの値も十分大きく設定しておく必要があります。

-w <Num>
  <Num>　　1 行あたりの半角文字数

-t <Num>
  <Num>　　トップ・マージン（単位は行数）

-b <Num>
  <Num>　　ボトム・マージン（単位は行数）

-i <Num>
  <Num>　　インデント（単位は桁数）

-x <Value>
  <Value>　　改行、復帰、タブ・コードの処理方法を示す数字

  <Value>　　は以下の内いずれか 1 つを指定します。

　　0　改行、復帰、タブ・コードはそのままプリンターに送られます。
　　1　復帰コードの後に改行コードが付けられてプリンターに送られます。
　　2　改行、垂直タブ・コードの後に復帰コードが付けられてプリンターに送られます。

-z <Value>
  <Value>　　プリンターの出力方法を示す数字
　　Canon LASER SHOT ではページのサイズと出力方法を指定する文字列に拡張されています。

  <Value>　　は以下の内いずれか 1 つを指定します。

　　0　出力方向を時計回りに 0 度回転させます。
　　1　出力方向を時計回りに 90 度回転させます。
　　2　出力方向を時計回りに 180 度回転させます。
　　3　出力方向を時計回りに 270 度回転させます。

Canon LASER SHOT では以下の指定も可能です。

A4	A4 ポートレート（デフォルト）
A4R	A4 ランドスケープ
A5	A5 ポートレート
A5R	A5 ランドスケープ
A6	はがきポートレート
A6R	はがきポートレート
B4	B4 ポートレート
B4R	B4 ランドスケープ
B5	B5 ポートレート
B5R	B5 ランドスケープ

-L <Bool>

<Bool>　　ページの幅より桁数の多い行についての処理方法を示す値

<Bool>　　は以下の内いずれか 1 つを指定します。

+　ページの右側より先の文字列は次の行のインデント位置から印字されます。（デフォルト）
!　ページの右側より先の文字列は印字されません。

-V <Bool>

<Bool>　　縦書きモード On/Off を示す値

<Bool>　　は以下の内いずれか 1 つを指定します。

+　縦書き印字
!　縦書き印字（デフォルト）

印字の属性に関するフラグ

-p <Num>

<Num>　　1 インチあたりの半角文字数

<Num>　　は IBM のプインターの日本語データ・ストリームでは以下の内いずれか 1 つを指定します。

10　　1 インチあたり 10 文字
12　　1 インチあたり 12 文字
13.4　　1 インチあたり 13.4 文字
15　　1 インチあたり 15 文字

<Num>　　は Proprinter データ・ストリームでは以下の内いずれか 1 つを指定します。

10　　1 インチあたり 10 文字
12　　1 インチあたり 12 文字

Canon LASER SHOT では、<Num>は任意の数字を指定できます。但し、指定した大きさのフォントがない場合には最もそれに近いものになります。

-K <Bool>

<Bool>　　横幅縮小文字モード On/Off を示す値

<Bool>　　は以下の内いずれか 1 つを指定します。

+　横幅縮小モード
!　通常モード（デフォルト）

-W <Bool>

<Bool>　　文字拡大モード On/Off を示す値

<Bool>　　は以下の内いずれか 1 つを指定します。

+　文字拡大モード（横幅が 2 倍になります）
!　通常モード（デフォルト）

-v <Num>

<Num>　1 インチあたりの行数

<Bool>　は IBM のプリンターの日本語データ・ストリームでは、以下の内いずれか 1 つを指定します。

2　1 インチあたり 2 行
3　1 インチあたり 3 行
4　1 インチあたり 4 行
5　1 インチあたり 5 行
6　1 インチあたり 6 行
7.5　1 インチあたり 7.5 行
8　1 インチあたり 8 行

<Num>は Proprinter データ・ストリームでは、以下の内いずれか 1 つを指定します。

6　1 インチあたり 6 行
8　1 インチあたり 8 行

Canon LASER SHOT では<Num>は任意の数字を指定できます。但し、指定した大きさのフォントがない場合には最もそれに近いものになります。

-E <Bool>

<Bool>　垂直方向の文字拡大 On/Off を示す値

<Bool>　は以下の内いずれか 1 つを指定します。

+　垂直方向の文字の大きさが 2 倍になります。
!　通常（デフォルト）

-q <Value>

<Value>　印字品質を示す数字

<Value>　は以下の内いずれか 1 つを指定します。

1　Draft Quality
2　Near Letter Quality

-S <Bool>

<Bool>　印字速度を示す値

<Bool>　は以下の内いずれか 1 つを指定します。

+　高速印字モードになります。
!　通常（デフォルト）

-U <Bool>

<Bool>　　片方向印字モード On/Off を示す値

<Bool>　　は以下の内いずれか 1 つを指定します。

+　片方向印字モード
!　通常モード（デフォルト）

-y <Bool>

<Bool>　　-　2 度打ちモード On/Off を示す値

<Bool>　　は以下の内いずれか 1 つを指定します。

+　2 度打ちモード
!　通常モード（デフォルト）

-e <Value>

<Value>　　印字の強調を示す値

<Value>　　は以下の内いずれか 1 つを指定します。

+　強調印字モード
!　通常モード（デフォルト）

Canon LASER SHOT では以下の値も使用可能です。

1 から 15 の数字　強調度を数字で示します。8 が通常で 15 が最も線が太くなり、1 が最も線が細くなります。

-F　[＜ユーザー定義ファイル＞]　[,［＜漢字ファイル＞]　[,　＜かなファイル＞]

＜ユーザー定義ファイル＞　ユーザー定義文字または IBM 拡張文字セット等を印字する場合、その文字の登録してあるフォント・ファイルの名前を絶対パス名で指定します。

/usr/lib/Xll/fonts/IBM_JPN17.pcf.Z、またはその他のユーザー定義ファイルを指定してください。

＜漢字ファイル＞、＜かなファイル＞は 4019 の場合に指定可能です。

＜漢字ファイル＞　/usr/lib/Xll/fonts/Kanji17.pcf.Z、またはその他の漢字ファイルを指定してください。

＜かなファイル＞　/usr/lib/Xll/fonts/Kana7.pcf.Z、またはその他のかなファイルを指定してください。

各ファイル名の間には‘,’を入れてスペースやタブは入れないでください。

-u <Value>

<Value>　　使用するトレーを示す値
<Value>　　は以下の内いずれか 1 つを指定します。

1　下トレーを使用します。
2　上トレーを使用します

プリント・ジョブに関するフラグ

-N <Num>
<Num>　　出力する部数

出力する部数を指定します。デフォルトは 1 部だけ出力されます。

-r
プリント・ジョブが終了後、そのファイルを削除します。

-c

プリントされるファイルのコピーを作り、コピーによって作られた方のファイルを出力します。プリント・ジョブを作動させ、そのジョブが終わる前にファイルを編集するような場合はこのフラグを使用します。

-g <Num>
<Num>　　出力を開始するページ番号

-n

プリント・ジョブの終了が通知されます。“-D”（出力結果を他のユーザーにも配布する為のオプション）が指定された場合、そのユーザーにもプリント・ジョブ終了が通知されます。

-m <Text>
<Text>　　メッセージ・テキストの指定

プリント・ジョブがプリンターに割り当てられ出力の用意が完了した時、指定したメッセージがコンソールに表示されます。

-M <File name>
<File name>　　メッセージ・テキストを含んだファイルの指定

上記の”-m”に似ていますが、”-m”オプションではメッセージをコマンドラインで直接指定するのに対して、”-M”オプションではメッセージを含んだファイル名を指定します。

-R <Num>
<Num>　　プリント・ジョブの優先度

<Num>には 1 から 30 の数字を指定します。数字の大きい方が優先度は高くなります。一般ユーザーが使用できる優先度は 1 から 20 までです。初期値は 15 に設定されています。

-C

プリント・ジョブによって出されたメッセージをディスプレイに表示する代わりに、そのユーザーに mail が送られます。

その他のフラグ

-a <Preview Option>
　<Preview Option>　オプションのプレビューの On/Off を示す数字
実際にプリンターを出力することなしに、各オプションの現在の設定値を見ることができます。<Preview Option>には以下の内いずれか 1 つを指定します。

0	通常の出力処理
1	フラグの現設定値のリストを表示します。

-A <Value>
　<Value>　診断出力のレベルを示す数字
<Value>には以下の内いずれか 1 つを指定します。診断出力には、プリント・ジョブの処理中に起きたエラー状況の出力及びプリント・ジョブからのステータス報告の出力が含まれます。

0	標準エラー出力への全ての出力を破棄します。
1	標準エラー出力がある場合、そのメッセージ及びエラー個所を表示します。
2	標準エラー出力がある場合、そのメッセージ及びエラー個所を表示します。標準エラー出力がない場合、印刷時に使用されたオプションと、そのオプションに対する値を全て表示します。

-B <Value>
　<Value>　ヘッダー、トレイラー・ページの有無を示す数字
出力結果にヘッダーページあるいはトレイラー・ページを付けるかどうかを指定します。<value>には以下の内 2 文字の組合せを指定し、1 文字目がヘッダーページに、2 文字目がトレイラー・ページに対応します。

a	常に付ける。
n	常に付けない。
g	1 回のプリント・ジョブに対して 1 ページだけ付ける。

-N <Name>
　<Name>　ヘッダー・ページに印刷されるホスト名

-Z <Bool>
　<Bool>　各ファイルを出力後、改ページするかどうか示す値
　<Bool>には以下の内いずれか 1 つを指定します。

+	改ページする。
－	改ページしない。

-O <Value>
　<Value>　使用する紙の種類を示す数字
　<Value>には以下の内いずれか 1 つを指定します。

1	手差しの単票
2	連続用紙
3	自動給紙装置付の単票

-j <Bool>
　<Bool>　プリンターの初期値の有無を指定
各ファイルを出力する前に、プリンターを初期化するかどうかを指定します。
<Bool>には以下の内いずれか 1 つを指定します。

+	初期化します。
－	初期化しません。

-T <Text>
　<Text>　プリント・ジョブのタイトル
　このオプションでタイトルを指定しない場合は、そのプリント・ジョブで指定した最初のファイル名がそのジョブのタイトルになります。プリント・ジョブはヘッダー・ページ上とキュー状態を調べる時に使用されます。

-D <User Name>
　<User Name>　　ヘッダー・ページに印刷するユーザー名

3.2.2 各プリンターの印刷ジョブ属性

印刷キューに設定した各プリンターの印刷ジョブ属性は、SMITを使用して変更／表示ができます。手順は次の通りです。

① SMIT を起動します。
② －印刷スプーリング
　　－プログラミング・ツール
　　　－プリンタ属性データベース（仮想プリンタ）の変更／表示を選択します。
③ 希望するプリンターにカーソルを移動して、ENTER を押します。
④ 指示に従って属性値を操作してください。

3.2.3 qprt による出力例

PC コードのファイル file1 をデフォルトのプリント・キューに出力する場合

　　$ qprt　file1

EUC コードのファイル file2 をデフォルトのプリント・キューに出力する場合

　　$ qprt　–XIBM–eucJP　file2

3.2.4 日本語ポストスクリプト・プリンターへの出力

ポストスクリプト・プリンターへ印刷を行う場合には、enscript コマンドでファイルをポストスクリプト形式に変換する必要があります。enscript コマンドを使用するには､以下のファイルセットをインストールしてください。

　　bos.txt.ts　　TranScript Tools

例）以下のコマンドは、テキスト・ファイル text をポストスクリプト形式のファイル text.ps に変換します。

　　enscript　　-ptext.ps　text

enscript コマンドの詳しい内容については、以下のマニュアルを参照してください。

AIX Version 6.1 Commands Reference, Volume 2　　SC23-5244

3.3 xpr の Canon LASER SHOT サポート

X ウィンドウのダンプイメージを Canon LASER SHOT プリンターに LIPS モードで出力できます。LIPS III 対応機種に出力の場合は、

```
$ xwd –xy | xpr –device lips3 –compact |qprt –Pcanon -dp
```

また、LIPS II 対応機種の場合は、

```
$ xwd –xy | xpr –device lips2 –compact |qprt –Pcanon –dp
```

と実行して下さい。

3.4 busy_delay の設定

使用するマシン・タイプとプリンターの機種によっては、特にpSeriesの最新機種に古いタイプのセントロニクスI/Fの遅いプリンターを接続した場合、正しく印刷が行えないことがあります。
このような場合には、busy_delay（文字間の遅延）の値を調節してください。

SMITで文字間の遅延の値を 10マイクロ秒に設定し、正しく印刷されるかどうかを確認します。
10マイクロ秒の設定で正しく印刷されない場合には、正しく印刷されるまで 10マイクロ秒ずつ増やしてください。

SMITの使用手順は次の通りです。

① SMIT を起動します。
② - 印刷スプーリング
　- プリンタ接続の特性の変更／表示
　- ローカル・プリンタ
　を選択します。
③ 文字間の遅延を入力して、ENTER を押します。

3.5 ウィンドウのイメージ印刷

本節では、AIXwindows のウィンドウ・イメージを日本語プリンターに出力する方法について解説します。

ウィンドウ・イメージをプリンターに出力するには、以下のコマンドを使用します。

xwd: ウィンドウのイメージ・データを作成します。
xpr: ウィンドウのイメージ・データを、プリンター固有なフォーマットに変換します。
qprt: ファイルをプリンターに出力します。

IBM 日本語プリンターを使用する際には、xpr において次のオプションが必要になります。

-device jprinter

使用例）

$ xwd –xy | xpr –device jprinter –plane 0 | qprt –dp –j+

1. xwd の-xy オプションにより、xwd は XYPixmap フォーマットのウィンドウイメージを作成します（省略時は Zpixmap フォーマット）。xpr は XYPixmap フォーマットのウィンドウイメージを受け取り-plane オプションにより任意の XYBitmap フォーマットのイメージを抽出し、そのイメージを元にプリンター固有なフォーマットに変換します。

xwd, xpr 及び qprt についての詳しい説明は、オンライン・マニュアルまたは下記マニュアルを参照して下さい。

AIX Version 6.1 Commands Reference, Volume 4 SC23-5246
AIX Version 6.1 Commands Reference, Volume 6 SC23-5248

注）
プリンターのイメージ・データ折り返しモードにより、イメージ・データが最大印字位置を超えた時に自動改行を実行し残りを印字するか、超えたデータを捨てるかを選択する事ができます。残りを印字するモードに選択した場合、印字結果が正常に出力されていない場合があります。イメージ・データ折り返しモードについては、各プリンターのマニュアルを参照して下さい。

xpr では-header 及び-trailer オプションによりそのウィンドウ・イメージにタイトルを挿入する事ができます。このタイトルには IBM-943 コード及び IBM-eucJP コードいずれも使用可能ですが、IBM-eucJP コードを使用する場合には、qprt を実行する際-dp オプションは使用できません（日本語プリンター自身は PC コードのみをサポートする為。）

IBM-eucJP コードを使用する際には ”qprt -XIBM-eucJP”のように-X オプションを指定して下さい。これにより qprt はイメージ中の IBM-eucJP コードを IBM-943 コードに変換してプリンターに出力します。

3.6 System V 印刷サービスにおける日本語プリンター・サポート

日本語データ・ストリーム PAGES、ESC/P、LIPS を扱う日本語プリンターのサポートを System V 印刷サービスにおいても提供します。ただし、ESC/P についてはサンプル機能です。

注) ESC/Pを使用する場合は、プリンターの条件等により、正しく印刷できない場合があります。

System V 印刷の詳細な情報は、以下のマニュアルを参照してください。

「AIX バージョン 6.1 プリンターおよび印刷」 (SC88-4592)　System V プリンターの構成

3.6.1 インストール

System V 印刷サービスを使用するには、以下のファイルセットをインストールしてください。

```
bos.svprint.fonts            System V Print Fonts
bos.svprint.ps               System V Print Postscript
bos.svprint.rte              System V Print Subsystem
bos.svprint.trans            System V Print Translation
bos.terminfo.svprint.data    System V Printer Terminal Definitions
```

また、AIX 印刷サービスのファイルセット

```
printers.rte                 Printer Backend
```

と、使用するデータ・ストリームの各ファイルセットもインストールしてください。

データ・ストリーム	ファイルセット
PAGES	printers.ibm-pages_JP.rte
ESC/P	printers.escpj84_JP.rte
LIPS4	printers.lips4_JP.rte
LIPS2+, LIPS3	printers.canlbp-A404F_JP.rte printers.canlbp-B406G.rte printers.canlbp.rte

3.6.2 設定手順

この System V 印刷サービスでは、AIX 印刷サービスで使用されている各データ・ストリームのコロン・ファイルによる属性定義データベースを利用することで、共通のフォーマッター・フィルターを使用しています。

はじめに、AIX 印刷での印刷キューの追加を行い、カスタマイズされた属性定義データベースを作成します。以下には、ローカル(パラレル・ポート)での IBM PAGES プリンターの設定例を示していきます。

コマンド例:

```
# /usr/lib/lpd/pio/etc/piomkpq -A local -p ibm-pages -v ibm-pages -s parallel -r ppa0 -D kji -q page -w p
```

印刷サブシステムを System V に変更し、デバイスの定義、プリンターの構成、フィルターの定義、プリンターの活動化、宛先の受入れ設定を順次行います。この設定例ではデバイス名に lp0、プリンター名(および宛先名)に page を指定しています。

コマンド例:
```
# switch.prt -s SystemV
# rmdev -d -l lp0
# mkdev -c printer -t opp -s parallel -p ppa0 -w p
# lpadmin -p page -v /dev/lp0 -h -m DS_JP -I simple -o nobanner -F beginning -T ds-jp
# lpfilter -f page -F /etc/lp/fd/DS_JP.fd
# enable page
# accept page
```

プリンターやフィルターの設定状況を表示するには、以下のコマンドが利用できます。

コマンド例:
```
# lpstat -lp page
# lpfilter -f page -l
```

日本語プリンター・サポートのために提供されている構成情報は以下のとおりです。

インターフェース・プログラム・オプション	型式(モデル)	DS_JP
	プリンター・タイプ	IBM PAGES Epson ESC/P Canon LIPS
	コンテンツ・タイプ	simple
terminfo データベースへのエントリー		ds-jp
フィルター定義記述ファイル		/etc/lp/fd/DS_JP.fd

3.6.3 使用方法

印刷要求を送信するには lp コマンドを使用します。日本語データ・ストリームを利用する場合に、System V 印刷サブシステムの lp コマンドで使用するフラグを説明します。

-d 宛先名
印刷を行うプリンターを選択します。ここに設定した日本語プリンター名を指定します。

-L ロケール名
この印刷要求で使用されるロケールとして、印刷するファイルのロケールを指定します。Ja_JP、ja_JP、または JA_JPが指定可能です。省略時は現在のロケールとなります。

-T コンテンツ・タイプ
フィルターの Input types で定義されているコンテンツ・タイプのリストからこの印刷要求で使用するデータ・ストリームを選択します。PAGES、ESCP、または LIPS が指定可能です。

-o オプション
lpi=2|3|4|5|6|7.5|8
行ピッチを設定します。qprt コマンドの-v オプションに対応します。
cpi=pica|elite
文字ピッチを設定します。pica は qprt コマンドの-p オプションの 10(10 文字/インチ)に、elite は 12(12 文字/インチ)に対応します。

注) その他のオプションについては、lsvirprt コマンドを使って属性定義データベースを変更してください。JISX0213 の文字を印刷する場合の Unicode フォント・ファイルの指定等はこの方法で行います。

- 以下に使用例を示します。

1. SJIS ファイル File1 を IBM PAGES プリンターに行ピッチと文字ピッチを指定して印刷します。
   ```
   lp -d page -L Ja_JP -T PAGES -o "cpi=elite lpi=4" File1
   ```

4. 共通デスクトップ環境での日本語入出力

■ 目次

4.1 はじめに

AIX 共通デスクトップ環境(CDE) は、AIX 4.1.3 またはそれ以降のバージョンで提供されるグラフィカル・ユーザ・インタフェースで、使いやすさを考慮して、直観的に環境の管理を行えるように設計されています。
共通デスクトップ環境は、これまで AIX で提供されていた AIXwindows 環境にかわって、主な UNIX ベンダー間で共通の操作環境を提供しています。

ここでは共通デスクトップ環境の基本的な使用方法について紹介します。

AIX 共通デスクトップ環境(CDE) に関連するマニュアルには以下のものがあります。
詳細については、それぞれのマニュアルを参照してください。

Common Desktop Environment 1.0: User's Guide
(SC23-2793, AIX 5L Version 5.1 Base Documentation CD)

Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide
(SC23-2795, AIX 5L Version 5.1 Base Documentation CD)

Common Desktop Environment 1.0: Internationalization Programmer's Guide
(SC23-2788, AIX 5L Version 5.1 Base Documentation CD)

Common Desktop Environment 1.0: Programmer's Overview
(SC23-2789, AIX 5L Version 5.1 Base Documentation CD)

Common Desktop Environment 1.0: Programmer's Guide
(SC23-2790, AIX 5L Version 5.1 Base Documentation CD)

4.2　デスクトップの概要

デスクトップには、作業を整理し管理するためのワークスペース、ウィンドウ、およびフロントパネルなどがあります。

ワークスペースは、作業に必要なウィンドウなどを配置する基本的な画面領域です。
ウィンドウは、ソフトウェア・アプリケーションを格納し、移動、サイズ変更、アイコン化してワークスペースに配置することができます。アプリケーションは、通常オブジェクトとしての操作、情報の入力を可能にするコントロールとして作成されています。
フロントパネルは、頻繁に使用する、それぞれのワークスペースで適用可能なコントロールの集合です。

4.3　共通デスクトップ環境の始動方法

以下に示す手順に従ってシステムにログインすると、共通デスクトップ環境が始動します。

1. ユーザー名を入力し、<Enter>キーを押すか、[了解]をクリックしてください。
2. パスワードを入力し、<Enter>キーを押すか、[了解]をクリックしてください。

ログイン画面のオプション・ボタンより表示されるメニューから、以下のようなセッション・オプションを変更することができます。

言　　語　**セッションで使用する言語環境を選択します。**
AIX 4.3.2 またはそれ以降のバージョンでは以下の日本語環境がサポートされています。

日本語(PC) IBM-943	シフト JIS コードで環境変数 LANG は Ja_JP です。
日本語(EUC) IBM-eucJP	日本語 EUC コードで環境変数 LANG は ja_JP です。
日本語 UTF-8	Unicode で環境変数 LANG は JA_JP です。

セッション　**以下のグラフィック環境を選択します。**

通常のデスクトップ	共通デスクトップ環境
復旧セッション	Ｘウィンドウ環境

注）日本語ロケール以外で日本語入力機能を使用する場合には、環境変数 LANG を日本語に設定するか lang オプション等を使ってアプリケーションを起動してください。

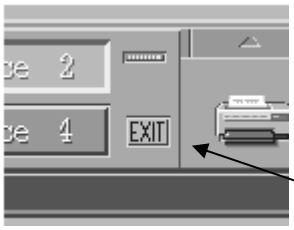
4.4　共通デスクトップ環境の終了方法

以下に示す手順に従って共通デスクトップ環境のセッションを終了します。

1. 実行中のアプリケーションがある場合、デスクトップからログアウトする前に、作業を保存するか、正しい手順に従って終了させます。

2. フロントパネルで、[終了]コントロールをクリックしてください。
   または、[ワークスペース・メニュー]で、[ログアウト]を選択してください。

   ① ポインタをワークスペースの背景の上に移動してください。
   ② マウス・ボタン 3 を押して、[ワークスペース・メニュー]を表示してください。
   ③ [ログアウト]までドラッグして、マウス・ボタンを離してください。



終了コントロール

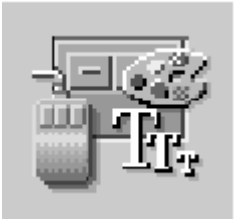
スタイル・マネージャーの起動により、ログアウトする際に現在のセッションを保存するか、ログアウト確認ダイアログを表示するかをカストマイズすることができます。

4.5　共通デスクトップ環境のカストマイズの方法

4.5.1　デスクトップの基本カストマイズ

共通デスクトップ環境は、スタイル・マネージャーを使って簡単にカストマイズすることができます。スタイル・マネージャーを起動するには、フロントパネルで[デスクトップスタイル]コントロールをクリックしてください。

デスクトップ・スタイル・コントロール



以下のようなスタイル・マネージャーが表示されます。

スタイル・マネージャーのメイン・ウィンドウには、以下の 10 個のコントロールがあります。これらをクリックして、カストマイズを行うためのダイアログ・ボックスを開いてください。

カラー	デスクトップ・カラーおよびパレットを選択します。
フォント	アプリケーション・フォント・サイズを選択します。
背景	ワークスペースの背景パターンを選択します。
キーボード	キーを押したときの音量、または文字反復機能を設定します。
マウス	マウス・ボタンのクリック設定、ダブルクリック間隔、ポインタ速度、ポインタ移動のしきい値を変更します。
ビープ音	ビープ音の音量、音程、音長を変更します。
画面	スクリーンセーバおよび画面ロック動作を指定します。
ウィンドウ	ウィンドウ・フォーカスの獲得方法、ウィンドウを手前にした場合に、どの時点でフォーカスを受領するか、およびどこにウィンドウ・アイコンを配置するかを指定します。
起動	セッションの開始方法および終了方法を指定します。
ワークスペース	ワークスペース・コントロールの動作を指定します。

4.5.2 フォント・サイズを変更するには

以下に示す手順に従ってスタイル・マネージャーにより、使用するフォントのサイズを変更することができます。

1. スタイル・マネージャーの [フォント] コントロールをクリックしてください。以下のようなダイアログ・ボックスが表示されます。

2. サイズを選択してください。プレビューに選択したサイズのフォントが表示されます。

3. [了解]をクリックしてください。

フォント・サイズの変更は、それぞれのアプリケーションが再起動された場合に、有効になります。

4.5.3 ユーザの .profile ファイルを使用するには

デフォルトでは、デスクトップにログインする際に読みこまれるソース・ファイルは$HOME/.profile ではありません。.profile ファイルをソースにするかどうかは、DTSOURCEPROFILE 環境変数が制御します。この変数の値を "true" に設定すると、このプロファイルがソースになります。それ以外の場合、このプロファイルはソースにはなりません。デフォルトではこの変数が未設定になっています。

以下に示す手順に従って DTSOURCEPROFILE 環境変数を設定してください。

1. 個人ユーザの場合

   $HOME/.dtprofile ファイルを編集して、DTSOURCEPROFILE=true がコメントされていないようにしてください。

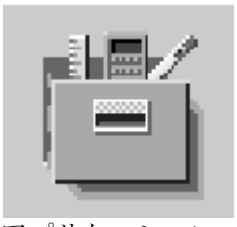
2. 全ユーザの場合

   /etc/dt/config/Xsession.d ディレクトリに DTSOURCEPROFILE=true を設定する実行可能形式の sh スクリプトまたは ksh スクリプトを作成してください。

4.5.4 aixterm を利用するには

共通デスクトップ環境では、通常はアプリケーションマネージャのデスクトップツールの中から aixterm を起動することができます。しかし、以下の方法で aixterm のアイコンを個人アプリケーションのサブパネルにコピーすることにより、フロントパネルから簡単に aixterm を利用することができます。

1. フロントパネルからアプリケーションマネージャをクリックして、起動してください。



アプリケーション・マネージャ

2. アプリケーションマネージャの中の[デスクトップツール]フォルダをダブル・クリックしてください。

3. フロントパネルの[テキストエディタ]上の上矢印をクリックして、[個人アプリケーション]サブパネルを開いてください。

4. [デスクトップツール]フォルダの中の aixterm アイコンをドラッグして、[個人アプリケーション]サブパネルの[アイコンのインストール]というアイコン上にドロップしてください。

4.6　端末エミュレーター (dtterm)

端末エミュレーター(dtterm)は、ウィンドウ上での UNIX コマンドの実行や、他のウィンドウなどとのテキストのカット&ペーストが使用できるアプリケーションです。

4.6.1　dtterm の起動方法

共通デスクトップ環境で dtterm を起動するには以下の方法があります。

フロントパネルから
既存の端末エミュレーター・ウィンドウからコマンドで
ファイル・マネージャーから
アプリケーション・マネージャーから
dtterm ウィンドウのプルダウン・メニューから

フロントパネルから dtterm を起動するには、[個人アプリケーション]サブパネルの中にある[端末エミュレーター]コントロールをクリックしてください。

また以下の手順で[端末エミュレータ]コントロールをサブパネルからフロントパネルのメインパネルに表示することもできます。

1. [個人アプリケーション]サブパネルを開いてください。
2. [端末エミュレータ]コントロール上でマウス・ボタン 3 を押して、ポップアップ・メニューを表示してください。
3. [メインパネルに表示]までドラッグして、マウス・ボタンを離してください。

既存の端末エミュレーター・ウィンドウのコマンド行から dtterm を起動する場合には、例のようにいろいろなオプションを入力することもできます。

例）
次のコマンドは、環境変数 LANG が ja_JP、preeditType が OffTheSpot で dtterm ウィンドウを起動します。

```
LANG=ja_JP dtterm -xrm "dtterm*preeditType: OffTheSpot"
```

4.6.2 dtterm の終了方法

以下に示す手順に従って dtterm を終了します。

1. コマンド行で exit と入力しリターン・キーを押してください。
   または、[ウィンドウ]メニューのプルダウン・メニューで[閉じる]を選択してください。

4.6.3 dtterm のいろいろな使い方

dtterm の詳細な使用方法については、以下の手順でオンライン・ヘルプを参照してください。

1. dtterm を起動します。
2. dtterm ウィンドウの[ヘルプ]メニューから[使い方]を選択してください。

このページは空白です。

5. 日本語 GUI アプリケーションの作成

■ 目次

5.1 はじめに

AIX Version 6.1 のXウィンドウは、X Window System Version 11, Release 7 (X11R7)の仕様に基づいており、OSF/Motif R2.1 に準拠したグラフィカル・ユーザー・インターフェース(GUI)です。また、AIX の提供する国際化機能 NLS(National Language Support)は、単一のシステム上での国際的な利用環境を実現しています。

国際化(I18N)とは、アプリケーション・プログラムをソースの変更や再コンパイルをしたりせず、異なる国の言語などに対応できるようにすることです。また地域化(L10N)とは、特定国の固有の機能に国際化されたアプリケーション・プログラムを適合させることです。例えば、国際化されたアプリケーション・プログラムでは、setlocale() 関数を呼び出し、特定の国の文字やその国に固有の処理を含みません。
このアプリケーション・プログラムの実行時には、環境変数 LANG を指定し、文字列やフォントなどをリソース・ファイルに設定することで地域化します。(5.2 参照)

National Language Support の詳細については、

「AIX バージョン 6.1 ナショナル・ランゲージ・サポートガイドおよびリファレンス」 (SC88-4576)

を参照してください。

AIXwindows のプログラミングに関するマニュアルには、以下のものがあります。

AIXwindows Programming Guide
Motif 2.1 Programmer's Guide
Motif 2.1 Programmer's Reference
(AIX 5L Version 5.1 Base Documentation CD)

OSF/Motif に関する日本語マニュアルには、以下のものがあります。

OSF/Motif スタイル・ガイド リリース１．２日本語入力機能	SC88-0018
OSF/Motif プログラマーズ・ガイド リリース１．２	SC88-0019
OSF/Motif プログラマーズ・リファレンス リリース１．２	SC88-0020
OSF/Motif ユーザーズ・ガイド リリース１．２	SC88-0021

この章では、AIXwindows の上で稼働する日本語アプリケーションの作成例を紹介します。

5.2 Motif サンプル・プログラム

5.2.1 作成手順

Motif は、X Window System に基づく GUI です。Motif アプリケーションは、X Window System におけるクライアント・サーバー間の基本的な対話の方法を提供する Xlib、Xlib の高水準インターフェースを提供する X ツールキット・イントリンシックス(Xt)、及び Motif ツールキットの 3 つのライブラリを使用します。Xt はユーザー・インターフェースの基本的なオブジェクトとなるウィジェットの管理や、様々な X イベントに対応して呼び出されるアプリケーション手続きであるコールバックの処理などを行います。Motif は Xt のウィジェットの基本型を使用して、より多目的なウィジェット・セットを提供しています。

国際化機能に準拠した Motif アプリケーションの、一般的な作成手順と関連する関数は以下の通りです。

1. 言語環境の確立	XtSetLanguageProc()
2. イントリンシックスの初期化	XtAppInitialize()
3. ウィジェットの生成	XtSetArg()
	XtCreateManagedWidget()
4. コールバック手続きの定義	XtAddCallback()
5. ウィジェットの表示	XtRealizeWidget()
6. イベント・ループへ入る	XtAppMainLoop()

Motif プログラミングの詳細については、
「OSF/Motif プログラマーズ・ガイド リリース 1.2」(SC88-0019)
を参照してください。

5.2.2 サンプル・プログラム

付録「H. プログラム・リスト immsample.c」に Motif ウィジェットを用いたサンプル・プログラムを示します。

このサンプル・プログラムでは、ロウカラム・ウィジェット上に、プッシュボタン・ウィジェットとテキスト・ウィジェットを配置しています。また、プッシュボタンを押したときにプログラムを終了させるコールバックを追加しています。

注）このサンプル・プログラムは AIX V4.1 または AIX V4.2 の Motif1.2、AIX V4.3 以降の Motif 2.1 で共通に動作します。

5.2.3 コンパイルとリンクの方法

コンパイルして以下のライブラリをリンクしてください。

```
cc -qdbcs -o immsample immsample.c -lXm -lXt -lX11
```

5.2.4 言語固有の機能の設定方法例

国際化されたアプリケーションは、言語や習慣に関して異なる条件を持つ、世界の各地域で利用できるよう適合させることができます。ここでは日本語環境で操作する場合の地域化の例を説明します。

環境設定方法

以下の２行を ksh で実行することにより、環境変数 LANG およびアプリケーションが使用するリソース・ファイルを指定する環境変数 XENVIRONMENT を設定することができます。例えば、リソース・ファイルとしてカレント・ディレクトリの”resJ”を使用する場合は、予め用意されていることを確認してください。

```
export LANG=Ja_JP
export XENVIRONMENT=`pwd`/resJ
```

リソース・ファイルの内容

Motif ウィジェットのリソースで以下のものを地域化しておきます。

フォント	フォントは文字を表示する時の書体を定義するもので、fontList リソースとして用意されています。ここには日本語フォントを指定する必要があります。
文字列	ラベル、テキスト文字列など、文字を表示する各ウィジェットに日本語の文字列を与えることができます。ここでは、PushButton ウィジェットの labelString リソース、Text ウィジェットの value リソース、ウィンドウ・マネージャー・タイトルの title リソースをそれぞれ指定します。
プリエディットのタイプ	AIX 日本語入力機能を使う時は、preeditType リソースを必要に応じで指定してください。このリソースは入力方式のスタイルを指定するもので、プリエディット文字列の表示する位置を変更します。

ファイル"resJ"の内容は以下の通りです。

```
Immsample*fontList: -ibm_aix-mincho-medium-r-normal--27-*:
Immsample*Button*labelString:  終了ボタン
Immsample*Text*value:  日本語テキスト(Ja_JP.IBM-943)
Immsample*title: immsample AIX ウィンドウ
Immsample*preeditType: OffTheSpot
```

以下にサンプル・プログラム"immsample"の実行例を示します。

5.3 Xlib サンプル・プログラム

5.3.1 サンプル・プログラム

付録「I. プログラム・リスト ximsample.c」に Xlib 関数を用いたサンプル・プログラムを示します。

このサンプルプログラムは、X11R7 でサポートされている XIM API を用いてテキスト入力を行う一例です。ウィンドウは、テキスト入力領域とステータス表示領域から成ります。四角いカーソルが表示されている場所に、入力したテキストが表示されます。テキスト出力は、X11R7 の XOM(X Output Method) API を用いて行います。終了する時は、ウィンドウ上で任意のマウス・ボタンを押してください。また、オプションの指定により任意のスタイルの入力方式を選択できます。

注）このサンプル・プログラムは AIX V4.3 以降の X11R6 以上で動作します。

5.3.2 コンパイルとリンクの方法

コンパイルして以下のライブラリをリンクしてください。

```
cc -qdbcs ximsample.c -o ximsample -lX11
```

5.4 Motif 2.1 の特徴

ここでは Motif 2.1 での主な新規機能について説明します。Motif 1.2 からの変更点についての詳細は、以下のマニュアルを参照してください。

AIX Version 4.3 Differences Guide (SG24-2014-00) Chapter 4. Graphics Enhancements 4.3 Motif Version 2.1

5.4.1 新しいウィジェット

Motif 2.1 では、以下のようなウィジェットが新たにサポートされています。

Container
: Main Window と同様にアプリケーションの中心となる作業領域を作成することができます。オブジェクト指向に適合した多様な表示、操作方法を提供しています。

Note Book
: 実際にノートの形をしたウィジェットです。タブやスクローラーを使って、ページを操作します。

Combo Box
: ドロップ・ダウン・リストとも呼ばれます。テキスト・フィールドの中に入力するか、または、その下にリストを表示して、その中から選択することができます。

Spin Box
: アロー・ボタンとラベルを組み合わせたウィジェットですが、左右の矢印を押すことにより数値が素早く増減し、簡単に希望の値を選ぶことができます。

Thermometer Scale
: スケール・ウィジェットに XmNslidingMode リソースがサポートされ、従来の XmSLIDER モードの指定に加えて XmTHERMOMETER が指定できるようになりました。
これを使ってメーター表示が行えます。

注）OSF/Motif 2.0 の CSText ウィジェットは、OSF/Motif 2.1 以降サポートされていません。

5.4.2 レンディション

Motif 1.2 では、コンパウンド・ストリングを取り扱うためのさまざまな XmString 関数が提供されていました。Motif 2.1 では、新たにレンディション(データ・タイプ XmRendition)がサポートされ、これらの API もより使いやすくなっています。

レンディションとは、フォント、色、タブなどのコンパウンド・ストリングに付加されるデータの集まりと、それを識別するタグから構成されています。
レンディションを使用するために、以下の API が提供されています。

レンディションの作成
- XmRenditionCreate()

レンディションの編集
- XmRenditionRetrieve()
- XmRenditionUpdate()

レンディションの削除
- XmRenditionFree()

それぞれのタグによって名付けられたレンディションの集まりが、レンダーテーブル(データ・タイプ XmRenderTable)を形成します。Motif 2.1 では、幾つかのウィジェットに XmNrenderTable リソースが追加されています。このリソースにレンダーテーブルを設定することにより、ウィジェットでレンディションが使用できるようになります。レンダーテーブルを取り扱う API としては、以下のものが提供されています。

レンダーテーブルの作成
- XmRenderTableAddRenditions()

レンダーテーブルの編集
- XmRenderTableGetRenditions()
- XmRenderTableRemoveRenditions()
- XmRenderTableCopy()
- XmRenderTableGetTags()

レンダーテーブルの削除
- XmRenderTableFree()

レンダーテーブルのポーティング
- XmRenderTableCvtFromProp()
- XmRenderTableCvtToProp()

レンディションとレンダーテーブルは、Motif 1.2 でのフォント・リストの使用方法を置き換えるものです。Motif 1.2 でサポートされていた XmFontList 関数は、バックワード互換のために残されていますが、Motif 2.1 で新たにアプリケーションを作成する場合には、これらの XmRendition/XmRender 関数を使用してください。

テキスト・ウィジェット(XmText)、テキストフィールド・ウィジェット(XmTextField)は、その中に表示するテキストとしてコンパウン・ストリングではなく普通の文字列を使用しています。したがって、レンディションの中のコンパウン・ストリングに固有なデータは使用されません。

5.4.3 XmRendition サンプル・プログラム

付録「J. プログラム・リスト mrsample.c」に XmRendition 関数を用いたサンプル・プログラムを示します。

このサンプル・プログラムでは、フォーム・ウィジェット上に、テキスト・ウィジェットとプッシュボタン・ウィジェットを配置しています。また、プッシュボタンを押したときにプログラムを終了させるコールバックを追加しています フォント、文字列などすべてのリソースは、プログラム内で設定されています。

注）このサンプル・プログラムは AIX 4.3 以降の Motif 2.1 で動作します。

5.4.4 サンプル・プログラムのコンパイルとリンクの方法

コンパイルして以下のライブラリをリンクしてください。

```
cc -qdbcs -o mrsample mrsample.c -lXm -lXt -lX11
```

以下にサンプル・プログラム"mrsample"の実行例を示します。

このページは空白です。

6. コード・コンバージョン・サポート

■ 目次

6.1 コンバーターの概要

AIX では、国際化機能(NLS) のひとつとして、コードセット間のデータ変換を行うためのコンバーターを提供しています。コードセットとは、文字やコントロール・コードのコードポイントへの割当てを定義したものです。データが異なるコードセットを使用する他のシステムに送られる場合、この変換が必要となります。例えば、ホスト・システムに接続する際には、AIX の日本語 EUC またはシフト JIS のデータをホストの EBCDIC に変換する場合があります。
AIX のコンバージョン・サポートは、コマンドラインからのコンバージョンと、アプリケーションからのコンバージョンを行なうために、以下の２つのインタフェースを提供しています。

iconv コマンド（6.3 参照）
iconv ライブラリ関数（6.4 参照）

ここではこれらの iconv コンバーターの使用方法について紹介しています。

AIX システムがサポートする標準のコードセット・コンバーターには、以下の２つのタイプがあります。

テーブル・コンバーター
コードの対応表を記述したテーブルを用いて、シングル・バイトのコードセット間のデータ変換を行います。各コンバーターのテーブル・ファイルは、/usr/lib/nls/loc/iconvTable ディレクトリに インストールされています。またファイルセット bos.loc.adt.iconv により、末尾が_src で終わるテーブルのソース・ファイルもインストールされます。

アルゴリズム・コンバーター
変換ルールを規定したアルゴリズムを用いたプログラムによって、データ変換を行います。マルチ・バイトのコードセット用のコンバーターは、すべてこのタイプです。各コンバーターのプログラム・ファイルは、/usr/lib/nls/loc/iconv ディレクトリにインストールされています。

これらのコンバーターでサポートされている日本語のコードセットには、以下のものがあります。

IBM-930	ホスト EBCDIC(カタカナ・モード)
IBM-939	ホスト EBCDIC(英数モード)
IBM-932	シフト JIS(78 年度バイナリ旧 JIS 並び)
IBM-943	Microsoft Windows 互換シフト JIS(83 年度バイナリ新 JIS 並び)
IBM-eucJP	日本語 EUC
IBM-udcJP-GL	ユーザー定義領域、IBM 拡張文字、NEC 選定文字、NEC の IBM 選定文字を GL(Graphic Left = 0x20 - 0x7f)にマッピング
IBM-udcJP-GR	ユーザー定義領域、IBM 拡張文字、NEC 選定文字、NEC の IBM 選定文字を GR(Graphic Right = 0xa0 - 0xff)にマッピング
JISX0201.1976-GL	JISX0201 半角カタカナを GL にマッピング
JISX0201.1976-GR	JISX0201 半角カタカナを GR にマッピング
JISX0208.1978-GL	JISX0208 JIS 第 1、第 2 水準(78 年度バイナリ旧 JIS 並び)を GL にマッピング
JISX0208.1978-GR	JISX0208 JIS 第 1、第 2 水準(78 年度バイナリ旧 JIS 並び)を GR にマッピング
JISX0208.1983-GL	JISX0208 JIS 第 1、第 2 水準(83 年度バイナリ新 JIS 並び)を GL にマッピング
JISX0208.1983-GR	JISX0208 JIS 第 1、第 2 水準(83 年度バイナリ新 JIS 並び)を GR にマッピング
UTF-8	Unicode（日本語 UTF-8、JA_JP ロケール）

各種コードセット間のコンバーターに加えて、ネットワーク環境でのコミュニケーションを目的としたインターチェンジ・コンバーターもサポートされています。インターチェンジ・コンバーターには、以下のようなタイプがあります。

ISO2022 7-bit コンバーター
: 7 ビットのメール・プロトコルで用いられる、ISO2022 で定義された 7 ビット標準インターチェンジ・フォーマットとのデータ変換を行います。コードセット名は、fold7 です。

ISO2022 8-bit コンバーター
: 8 ビットのメール・プロトコルで用いられる、ISO2022 で定義された 8 ビット標準インターチェンジ・フォーマットとのデータ変換を行います。コードセット名は、fold8 です。

Compound Text コンバーター
: Xコンソーシアムによって定義された、Xクライアント間の通信で用いられるコンパウンド・テキストとのデータ変換を行います。コードセット名は、ct です。

uucode コンバーター
: uuencode および uudecode コマンドでのマッピングと同様のデータ変換を行います。コードセット名は、uucode です。

Unicode UCS-2 コンバーター
: Unicode として知られる ISO10646 標準の 16 ビット形式のコードセットである UCS-2 とのデータ変換を行います。コードセット名は、UCS-2 です。

Unicode UTF-32 コンバーター
: Unicode として知られる ISO10646 標準の 32 ビット形式のコードセットである UCS-4 とのデータ変換を行います。コードセット名は、UTF-32 です。

Unicode UTF-8 コンバーター
: X/Open によって定義された、UCS-2 をバイト列として符号化するためのフォーマットである UTF-8 とのデータ変換を行います。コードセット名は、UTF-8 です。

注) IBM-eucJP(日本語 EUC)および UTF-8(Unicode)は 83 年度新 JIS バイナリ並びです。

6.2　日本語サポート・コンバーター

以下の表に使用可能な日本語コードセットの組み合わせを示します。

FROM \ TO	IBM-930	IBM-939	IBM-932	IBM-943	IBM-eucJP	IBM-udcJP-GL	IBM-udcJP-GR	JISX0201.1976-GL	JISX0201.1976-GR	JISX0208.1978-GL	JISX0208.1978-GR	JISX0208.1983-GL	JISX0208.1983-GR	fold7	fold8	ct	uucode	UCS-2	UTF-32	UTF-8
IBM-930		—	○	○	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
IBM-939	—		○	○	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
IBM-932	○	○		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
IBM-943	○	○	○		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
IBM-eucJP	○	○	○	○		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
IBM-udcJP-GL	—	—	○	○	○		—	—	—	—	—	—	—	—	—	—	—	—	—	—
IBM-udcJP-GR	—	—	○	○	○	—		—	—	—	—	—	—	—	—	—	—	—	—	—
JISX0201.1976-GL	—	—	○	○	○	—	—		—	—	—	—	—	—	—	—	—	—	—	—
JISX0201.1976-GR	—	—	○	○	○	—	—	—		—	—	—	○	○	○	○	—	○	○	○
JISX0208.1978-GL	—	—	○	○	○	—	—	—	—		—	—	—	—	—	—	—	—	—	—
JISX0208.1978-GR	—	—	○	○	○	—	—	—	—	—		—	—	—	—	—	—	—	—	—
JISX0208.1983-GL	—	—	○	○	○	—	—	—	—	—	—		—	—	—	—	—	—	—	—
JISX0208.1983-GR	—	—	○	○	○	—	—	—	○	—	—	—		○	○	○	—	○	○	○
fold7	—	—	○	○	○	—	—	—	○	—	—	—	○		○	○	—	○	○	○
fold8	—	—	○	○	○	—	—	—	○	—	—	—	○	○		○	—	○	○	○
ct	—	—	○	○	○	—	—	—	○	—	—	—	○	○	○		—	○	○	○
uucode	—	—	○	○	○	—	—	—	—	—	—	—	—	—	—	—		—	—	○
UCS-2	—	—	○	○	○	—	—	—	○	—	—	—	○	○	○	○	—		○	○
UTF-32	—	—	○	○	○	—	—	—	○	—	—	—	○	○	○	○	—	○		○
UTF-8	—	—	○	○	○	—	—	—	○	—	—	—	○	○	○	○	○	○	○	

6.3 iconv コマンド

iconv コマンドのシンタックスを示します。

```
iconv -f FromCode -t ToCode [ FileName... ]
```

フラグの説明

-f FromCode	入力データのコードセットを指定します。
-t ToCode	出力データのコードセットを指定します。
FileName	変換される入力ファイルを指定します。

iconv コマンドは、標準入力から読みこまれる文字または指定されたファイルをコード変換し、標準出力に書き込みます。入力データは、FromCode で指定されたコードセットの文字でなければなりません。

例 1: 以下のコマンドは、AIX の日本語 EUC コードのデータを ISO2022 7-bit 形式に変換します。

```
iconv -f IBM-eucJP -t fold7 mail.local > mail.net
```

例 2: 以下のコマンドは、AIX の Ja_JP 環境で作成されたデータを JA_JP 環境用に変換します。

```
iconv -f IBM-943 -t UTF-8 text.Ja > text.JA
```

iconv コマンドのさらに詳しい内容については、以下のマニュアルを参照してください。

AIX Version 6.1 Commands Reference, Volume 3　　SC23-5245

注）AIX 日本語環境拡張キットには、日本語データ・コンバーターが含まれています。この conv コマンドを使用する場合にはデータ変換（マッピング）情報のカスタマイズが可能です。

6.4 iconv ライブラリ関数

iconv のライブラリ関数は、次の３つの API(アプリケーション・プログラミング・インターフェース)から成ります。これらの API は、X/Open XPG4 によって標準化されているものです。

iconv_open()
iconv()
iconv_close()

次にそれぞれの関数の使用方法をを示します。これらの関数は、iconv ライブラリ libiconv.a に含まれています。また、これらの関数を使用する場合は、ヘッダー・ファイル iconv.h をインクルードします。

各 iconv ライブラリ関数の詳細な内容については、以下のマニュアルを参照してください。

AIX Version 6.1 Technical Reference: Base Operating System and Extensions, Volume 1　　SC23-6608

6.4.1 iconv_open 関数

シンタックス

```
iconv_t iconv_open (ToCode, FromCode)
const char *ToCode, *FromCode;
```

パラメーターの説明

ToCode	変換後のコードセットを指定します。
FromCode	元のコードセットを指定します。

iconv_open 関数は、指定されたコードセットの変換を行うコンバーターを検索して、初期化し、オープンしたコンバーターの変換ディスクリプタを返します。初期化に失敗した場合には、-1 を返します。

6.4.2 iconv 関数

シンタックス

```
size_t iconv (CD, InBuf, InBytesLeft, OutBuf, OutBytesLeft)
iconv_t CD;
char **InBuf, **OutBuf;
size_t *InBytesLeft, *OutBytesLeft;
```

パラメーターの説明

CD	使用するコンバーターの変換ディスクリプタを指定します。
InBuf	変換する文字列を含むバッファーのポインターを指定します。
InBytesLeft	変換する文字列のバイト数をセットした変数へのポインターを指定します。
OutBuf	変換後用のバッファーのポインターを指定します。
OutBytesLeft	変換後用のバッファーの大きさをセットした変数へのポインターを指定します。

iconv 関数は、InBuf で指定された文字列を変換し OutBuf に指定されたバッファーに格納します。CD は、iconv_open() によって返された変換ディスクリプタでなければなりません。変換する文字列が全て正しく変換された場合、iconv()は 0 を返します。
また、InBuf および OutBuf は変換した文字の分だけ移動し、InBytesLeft および OutBytesLeft の変数の値から、そのバイト数が引かれます。変換に失敗した場合も、パラメーターは同様に変更され、iconv()は変換されなかった残りの文字列のバイト数、すなわち InBytesLeft の変数の値を返します。

6.4.3 iconv_close 関数

シンタックス

```
int iconv_close (CD)
iconv_t CD;
```

パラメーターの説明

CD　クローズするコンバーターの変換ディスクリプタを指定します。

iconv_close 関数は、iconv_open()によってオープンされたコンバーターをクローズします。

6.5　サンプル・プログラム

以下に iconv 関数を用いたサンプル・プログラムを示します。このサンプル・プログラムは、
ISO2022 7-bit 形式のデータを現在利用している AIX の日本語環境のデータに変換します。

例)　"cnvsample.c"

```
/*
 * sample program - cnvsample.c
 */
#include <stdio.h>
#include <locale.h>
#include <nl_types.h>
#include <langinfo.h>
#include <iconv.h>
#include <errno.h>

#define FROMCODE  "fold7"
#define BUFSIZE   256

extern int errno;

main(int argc, char **argv)
{
    FILE        *fp;
    char        *toCode;
    iconv_t     cd;
    const char  *inp;
    char        *outp;
    size_t      in_len, out_len;
    char        inbuf[BUFSIZE], outbuf[BUFSIZE];
    int         ret;

    setlocale(LC_ALL, "");

   /*
    *  open input file
    */
    if(argc != 2) {
        fprintf(stderr, "Syntax: cnvsample FileName¥n");
        exit(1);
    }
    else {
        fp = fopen(argv[1], "r");
        if(fp == NULL) {
            fprintf(stderr, "Error: Can't open input file.¥n");
            exit(1);
        }
    }

   /*
    *  open converter
    */
    toCode = nl_langinfo(CODESET);
    if((cd = iconv_open(toCode, FROMCODE)) == (iconv_t) -1) {
        fprintf(stderr, "Error: iconv_open failed.¥n");
        exit(1);
    }
    fprintf(stderr, "Executing iconv from %s to %s¥n", FROMCODE, toCode);

   /*
    *  read each line from input file until EOF
```

```
    */
    while(fgets(inbuf, BUFSIZE, fp) != NULL) {
       /*
        *  set parameters and convert
        */
        bzero(outbuf, BUFSIZE);
        inp = inbuf;
        in_len = strlen(inbuf);
        outp = outbuf;
        out_len = BUFSIZE;
        ret = iconv(cd, &inp, &in_len, &outp, &out_len);

        if(ret == 0) {
            fprintf(stdout, "%s", outbuf);
        }
        else {
           /*
            *  check iconv error
            */
            switch(errno) {
            case EILSEQ:
                fprintf(stderr, "Error: invalid input.¥n");
                break;
            case E2BIG:
                fprintf(stderr, "Error: buffer overflow.¥n");
                break;
            case EINVAL:
                fprintf(stderr, "Error: input incomplete.¥n");
                break;
            default:
                fprintf(stderr, "Error: iconv failed.¥n");
            }
            exit(1);
        }
    }

   /*
    *  close file and converter
    */
    fclose(fp);
    iconv_close(cd);
}
```

コンパイルして以下のライブラリをリンクしてください。

```
cc -o cnvsample cnvsample.c -liconv
```

実行例:

```
cnvsample mail.net > mail.local
```

注) このサンプル・プログラムは AIX バージョン 4 以降で共通に動作します。詳細なエラーチェックを行う場合は、
このサンプルのように errno をチェックしてください。

6.6 fold7 コンバーター

AIX の fold7 コンバーターはコード・セットを JIS7 ビット情報交換用符号に変換します。
この JIS7 ビット情報交換用符号は MSB がオフ（最上位ビットが 0）になっていて、ネットワーク上でデータを交換する時に有効です。
AIX の fold7 コンバーターは次のようにコード変換します。

文字の種類	fold7 コンバーターで変換後のコード・セット
ASCII（半角英数字） 半角カナ文字	JIS7 ビット情報交換用符号　X0201-1976
全角文字	JIS7 ビット情報交換用漢字符号　X0208-1983

それぞれの文字は次のようにエスケープ・シーケンスが付けられます.

バイト	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ASCII（半角英数字）	ESC	2/8	4/2													
半角カナ文字	ESC	2/8	4/9													
全角文字	ESC	2/4	4/2													
ユーザー定義文字	ESC	2/5	2/15	3/2	M	L	6/9	6/2	6/13	2/13	7/5	6/4	6/3	4/10	5/0	0/2

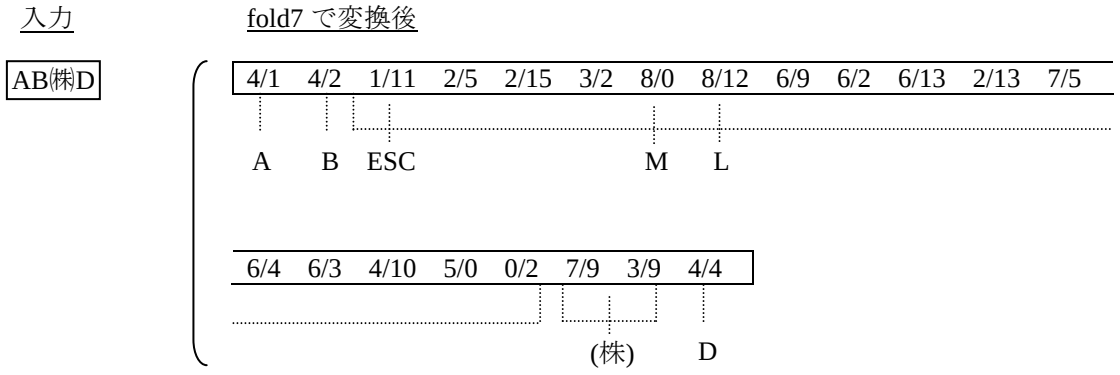
この後に続くユーザー定義文字の長さ（バイト数）を表します。
長さは次のように求められます。
長さ＝((M－128)＊128)＋(L－128)
（この長さにはML に続く 6－16 バイト分も含まれます。）

ESC はエスケープ・コード 1/11 を表しま

このコンバーターの初期状態は ASCII です。つまり最初の入力文字が ASCII の場合には変換された文字列はエスケープ・シーケンス（ESC 2/8 4/2）はつけられません。

ユーザー定義文字が来た場合には、現在の状態がセーブされ、ユーザー定義文字変換後に元の状態に戻されます。従って、ユーザー定義文字の前後の文字が同じ種類の場合には、ユーザー定義文字の後にはエスケープ・シーケンス・コードはつけられません。

例）



注）fold コンバーターとエスケープ・シーケンス

iconv_open コール後の iconv のデータ処理において、エスケープ・シーケンスが出力されるのは文字の種類が変更された場合のみです。同一コンバーター・ディスクリプターを使用していて同じ種類の文字がデータとして出力される場合には、iconv が複数回コールされてもその最初の iconv コールの時にのみエスケープ・シーケンスが出力されます。

iconv_open コール後の最初のデータ出力が ASCII 文字の場合は、エスケープ・シーケンスは出力されません。デフォルト値が ASCII 文字の為です。

0x1f 以下の値を持つコードの入力データに対しては、iconv はそのコードを制御コードとみなしてエスケープ・シーケンスなしでそのまま出力します。

6.7 fold8 コンバーター

fold8 コンバーターは、コードセットをJIS8ビット情報交換用符号に変換します。
以下にfold8 コンバーターとfold7 コンバーターで付加されるエスケープ・シーケンスの違いを示します。

- fold8 コンバーター

エスケープ・シーケンス	スタンダード・コードセット
01/11 02/04 04/00	GL JIS X0208.1978-0
01/11 02/09 04/09	GR JIS X0201.1976-1 の右半分
01/11 02/04 02/09 04/02	GR JIS X0208.1983-1
01/11 02/04 02/09 04/00	GR JISX0208.1978-1
01/11 02/05 02/15 03/02 M L 04/09 04/02 04/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02	GR IBM-udcJP(ユーザー定義文字) の右半分
01/11 02/08 04/09	GL JIS X0201.1976-0 の右半分
01/11 02/08 04/10	GL JIS X0201.1976 の左半分
01/11 02/04 02/08 04/02	GL JIS X0208.1983-0
01/11 02/04 04/02	GL JIS X0208.1983-0
01/11 02/04 04/00	GL JIS X0208.1978-0
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02	GL IBM-udcJP(ユーザー定義文字)

注) fold8にデータ変換されるとき付加されるエスケープ・シーケンスは、上の表の順となります。例えば、JISX0208.1983-0 の文字には、01/11 02/04 02/08 04/02 が使われます。

- fold7 コンバーター

エスケープ・シーケンス	スタンダード・コードセット
01/11 02/04 04/00	GL JIS X0208.1978-0
01/11 02/08 04/09	GL JIS X0201.1976-0 の右半分
01/11 02/08 04/10	GL JIS X0201.1976 の左半分
01/11 02/04 04/02	GL JIS X0208.1983-0
01/11 02/04 02/08 04/02	GL JIS X0208.1983-0
01/11 02/04 02/08 04/00	GL JISX0208.1978-0
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02	GL IBM-udcJP(ユーザー定義文字)

注) fold7にデータ変換されるとき付加されるエスケープ・シーケンスは、上の表の順となります。例えば、JISX0208.1983-0 の文字には、01/11 02/04 04/02 が使われます。

6.8 コードセット IBM-943

6.8.1 IBM-943 の概要

AIX V4.3.2 より、IBM のコードページ 943 に基づく新たな日本語 SJIS コードとして、コードセット IBM-943 がサポートされています。IBM-943 は、Microsoft Windows 95、98、および NT に採用されている日本語文字セットと互換性をもちます。例えば、これまでの日本語 PC コードのコードセット IBM-932 の場合、AIX で作成したファイルを Windows で使用すると新 JIS と旧 JIS の文字表示が入れ替わったり、Windows で作成したファイルを AIX で使用すると NEC 選定文字が表示されないなどの問題がありました。
IBM-943 では、このような文字も AIX と Windows で等しく取り扱われます。IBM-943 のサポートに伴ない、日本語ロケール Ja_JP のデフォルトのコードセットが、IBM-932 から IBM-943 に変更されました。従来から AIX でサポートされていた IBM-932 との違いは、以下の通りです。

IBM-943 の IBM-932 との相違点	文字数
JISX0208-1983 で行なわれた字形の入れ替えへの対応 **1	52 文字
JISX0208-1990 で追加された文字の割り当ての変更 **2	2 文字
JISX0208-1990 で行なわれた字形の変更への対応 **3	2 文字
JISX0208-1983 で追加された文字のうち IBM-932 ですでに IBM 非漢字拡張文字セットとして定義されていた文字の追加 **4	2 文字
NEC 選定文字セットの追加	83 文字
NEC の IBM 選定文字セットの追加	374 文字

**1 これらの 52 文字と IBM-943 のコードポイントを以下に示します。
IBM-932 でのコードポイントは、左右が入れ替わったものです。
IBM-943 は新 JIS 並び(1983 年度 JIS)対応、IBM-932 は旧 JIS 並び(1978 年度 JIS)対応です。

鰺	88B1	鰺	E9CB
鴬	89A7	鶯	E9F2
蛎	8A61	蠣	E579
撹	8A68	攪	9D98
竈	8A96	竃	E27D
潅	8AC1	灌	9FF3
諌	8AD0	諫	E67C
頚	8C7A	頸	E8F2
砿	8D7B	礦	E1E6
蕊	8EC7	藥	E541
靭	9078	靱	E8D5
賎	9147	賤	E6CB
壷	92D9	壺	9AE2
砺	9376	礪	E1E8
梼	938E	檮	9E8D
涛	9393	濤	9FB7
迩	93F4	邇	E78E
蝿	9488	蠅	E5A2
桧	954F	檜	9E77
侭	9699	儘	98D4
薮	96F7	藪	E54D
篭	9855	籠	E2C4
尭	8BC4	堯	EA9F
槙	968A	槇	EAA0
遥	9779	遙	EAA1
瑶	E0F4	瑤	EAA2

**2 これらの 2 文字と IBM-943 のコードポイントを以下に示します。
IBM-932 でのコードポイントは、左右が入れ替わったものです。

熙 E086 熙 EAA4

**3 これらの 2 文字と IBM-943 のコードポイントを以下に示します。
IBM-932 でのコードポイントは、左右が入れ替わったものです。

昂 8D56 昂 FAD0

**4 これらの 2 文字と IBM-943 のコードポイントを以下に示します。
括弧内は IBM-932 でのコードポイントです。

￢ 81CA (FA54)
∵ 81E6 (FA5B)

NEC 選定文字セットと NEC の IBM 選定文字セットの各文字については、以下のマニュアルを参照してください。

「AIX 5L 日本語コード一覧表」 (SC88-0427)

6.8.2 同一文字の処理

IBM-943 では、以下のような同一字形の文字が複数のコードポイントに存在します。

NEC 選定文字と IBM 拡張文字で定義されている 13 文字 **5
NEC 選定文字と JIS で定義されている 8 文字 **5
NEC 選定文字と IBM 拡張文字と JIS で定義されている 1 文字 **6
NEC の IBM 選定文字と IBM 拡張文字で定義されている 373 文字 **5
NEC の IBM 選定文字と IBM 拡張文字と JIS で定義されている 1 文字 **6

**5 これらの文字とコードポイントについては、
「AIX5L 日本語コード一覧表」(SC88-0427) の
NEC 選定文字セットと NEC の IBM 選定文字セットの項を参照してください。

**6 これらの文字の各コードポイントを以下に示します。

	NEC 選定文字	NEC の IBM 選定文字	IBM 拡張文字	JIS
∵	879A	なし	FA5B	81E6
￢	なし	EEF9	FA54	81CA

IBM-943 でこれらの文字が選択される優先順位は、以下の通りです。

1. JIS で定義されていれば、そのコードポイント
2. IBM 拡張文字で定義されていれば、そのコードポイント
3. NEC 選定文字で定義されていれば、そのコードポイント

例えば、iconv 等でこれらの文字が他のコードセットから IBM-943 に変換される場合、上記の優先順位が適用されます。ファイル名や vi では内部的に iconv 変換が行われているため、NEC 選定文字や NEC の IBM 選定文字は、自動的に JIS や IBM 拡張文字で定義されているコードポイントに変わります。

6.8.3 IBM-932 との互換性

AIX V4.3.1 以前の Ja_JP ロケール、すなわちコードセット IBM-932 で作成されたファイルが「1. IBM-943 の概要」で述べたコードポイントが入れ替わった文字を含んでいる場合、IBM-943 ではそれらの文字が違って表示されることになります。AIX V4.3.1 以前と同様の文字を表示させたい場合には、以下のように iconv コマンドを使ってファイルを変換

してください。

例）
iconv -f IBM-932 -t IBM-943 File.old > File.new

6.8.4 既存アプリケーションの考慮点

IBM-943 では、IBM-932 で未定義であった以下のコードポイントに新たに文字が定義されています。

81CA および 81E6
8740 から 87FC の NEC 選定文字の範囲
ED40 から EEFC の NEC の IBM 選定文字の範囲

また「1. IBM-943 の概要」で述べたように、IBM-932 とは字形とコードポイントの対応が変わっている文字があります。プログラム中でこれらのコードポイントに対し特別な処理を行っている場合、あるいは、iconv 関数を IBM-932 で使用している場合など、IBM-943 の Ja_JP ロケールでは正しく文字が扱われない可能性があります。このようなときには、プログラムの修正が必要となります。

注）AIX V5.1 以降においても従来の IBM-932 環境を利用したい場合には、Ja_JP.IBM-932 ロケールを使うことができます。（ただし、アプリケーションがロングロケール名に対応している必要があります。）
また、将来の AIX のリリースにおいては、IBM-932 環境のサポートはなくなる予定です。

例）
export LANG=Ja_JP.IBM-932

6.8.5 その他

ユーザー辞書ユーティリティー (kjdict)
AIX V4.3.1 以前に作成されたAIX日本語処理機能のユーザー辞書に、「1. IBM-943の概要」で述べたコードポイントが入れ替わった文字が含まれている場合、IBM-943の環境では違って表示されることになります。
AIX V4.3.1 以前と同様の文字を表示させたい場合には、新たに登録し直す必要があります。

フォント・ユーティリティー (fontutil)
IBM-943で新たに追加された、NEC選定文字セット、NECのIBM選定文字セットを使用する場合には、Extension メニューのコードセットから Internal_16を選択してください。

7. 日本語ユニコード・サポート

■ 目次

ここでは、AIX V4.3.2 より新たにサポートされた Unicode の日本語ロケール JA_JP について紹介します。

7.1 コードセットの概要

ある言語で必要とされる文字の集まりを定義したものをキャラクター・セットといいます。ひとつのキャラクター・セットは、異なるエンコーディングをもつことができます。例えば、日本語 EUC コードとシフト JIS コードでは、同じ漢字の文字が別々のコードポイントに割当てられています。コードセットとは、エンコーディングも含めたキャラクター・セットの定義といえます。すなわち、コードセットはコードポイントを定めた文字の集まりです。

AIX では、各言語の UTF-8 コードセットによる Unicode サポートが提供されています。Unicode というのは、ISO10646 として定義されている国際文字セットとその符号化方式の名称で、UCS-4 と UCS-2 の二つの形式があります。
UTF とは、これら UCS の Transformation Format と呼ばれるもので、バイト列として符号化するためのフォーマットです。
UTF-8 は、多国語言語環境を実現する UTF のひとつのコードセットです。AIX の UTF-8 コードセットでは、欧州単一通貨のユーロ記号もサポートしています。

7.2 UCS-2

UCS(Universal Coded Character Set)とは、ひとつのコードセットで世界の主要な言語をすべて表現することを目的とした、Unicode として知られる ISO10646 標準の名称です。32 ビット形式の UCS-4(4-octet form of the ISO10646 Universal Multiple-Octet Coded Character Set)は、400 億を超える文字を含むエンコーディング
を実現することができます。これに対し、16 ビット形式の UCS-2(2-octet form of the ISO10646 Universal Multiple-Octet Coded Character Set)は、BMP(Basic Multilingual Plane)と呼ばれる 0x0000 から 0xFFFF までのコードの値を表現でき、主要な世界標準で提供されているすべての文字を含んでいます。

AIX では、32 ビット環境のプロセス・コード(内部コード)として UCS-2 コードセットをサポートしています。また、AIX V5.3 からは 64 ビット環境のプロセス・コードとして UCS-4(コードセット名は UTF-32)をサポートしています。

AIX は、X Window System において Unicode の文字列を表示するための UnicodeTrueType Fonts を提供していますが、このフォントのエンコーディングは、BMP の範囲内では UCS-2 が使われています。また、Extension B(32 ビット)の範囲内では UCS-4 が使われています。

Unicode について詳しく知りたい方は、以下の Unicode Consortium の Web サイトを参照してください。

URL: http://www.unicode.org/

7.3 UTF-8

16 ビット形式の UCS-2 は、バイト単位で扱われるテキスト・ファイルなどで使用するには適しません。そこで AIX は、ファイル・コード(外部コード)として使用するための UTF-8 コードセットをサポートしています。これは、X/Open によって開発された File System Safe UCS Transformation Format(FSS-UTF)と呼ばれるもので、UCS-2 をバイト列として符号化したフォーマットです。

UTF-8 には、以下のような特徴があります。

ASCII(8bit)文字は 1 バイトで ASCII と同じコードポイントです。
ASCII 文字以外を表すマルチバイトの文字に、ASCII のコードポイントは現れません。
簡単に UCS との変換が行えます。
マルチバイト・コードの第一バイトはそれに続くバイト数を示し、また、第一バイトと同じものは残りのバイトに現れません。

UTF-8 の構成は以下の表の通りです。
シングルバイト 0 から 0x7f は、ASCII 文字用のです。
UCS の値は、マルチバイトの x ビットを合わせたものです。

UTF-8 Multibyte Codes				
Bytes	Bits	Hex Minimum	Hex Maximum	Byte Sequence in Binary
1	7	00000000	0000007F	0xxxxxxx
2	11	00000080	000007FF	110xxxxx 10xxxxxx
3	16	00000800	0000FFFF	1110xxxx 10xxxxxx 10xxxxxx

AIX では、以下ようなの各国言語の UTF-8 コードセットによる Unicode ロケールがサポートされています。

Lang.UTF-8

ただし、Lang はすべて大文字の既存のロケール名です。例えば、日本語では、

JA_JP.UTF-8

です。また、短縮ロケール名は、JA_JP です。

各国言語の Unicode ロケールとは、多国語言語環境である UTF-8 コードセットにおいて、その国の言語環境が使用可能であることを意味します。例えば JA_JP ロケールの場合、UTF-8 コードセットの日本語文字列が表示でき、デフォルトの Input Method として日本語入力機能が使用されます。日本語入力機能で使用できる文字は、AIX V5.1 では Ja_JP(IBM-943)と同じですが、AIX V5.2 以降は JISX0213 もサポートされます。（詳しくは「第 8 章 JISX0213 サポート」を参照してください。）

多国語言語環境を実現するには、使用する各国言語のロケール、および Unicode ロケールをインストールしてください。例えば、日本語と韓国語をインストールした場合、JA_JP ロケールでは日本語の、KO_KR ロケールでは韓国語の Input Method が、それぞれデフォルトとして使用され、どちらのロケールにおいても日本語および韓国語の文字列が表示できます。また「7.5.2 入力方法」で示す方法により、日本語も韓国語も入力が可能になります。

7.4 コンフィギュレーション

Unicode の日本語ロケール JA_JP.UTF-8 を使用するには、以下のファイルセットをインストールしてください。

```
X11.fnt.ucs.ttf           AIXwindows Unicode TrueType Fonts
X11.loc.JA_JP.Dt.rte      AIX CDE Locale Configuration - Japanese (UTF)
X11.loc.JA_JP.base.lib    AIXwindows Client Locale Config - Japanese (UTF)
X11.loc.JA_JP.base.rte    AIXwindows Locale Configuration - Japanese (UTF)
bos.iconv.ucs.com         Unicode Base Converters for AIX Code Sets/Fonts
bos.loc.com.utf           Common Locale Support - UTF-8
bos.loc.utf.JA_JP         Base System Locale UTF Code Set - Japanese
```

AIX V5.3 以降では、以下のファイルセットもインストールしてください。

```
X11.fnt.ucs.ttf_extb      AIXwindows Unicode TrueType Fonts - Extension B
```

AIX V5.1 のファイルセット X11.fnt.ucs.ttf には、JA_JP ロケールの共通デスクトップ環境で利用できる、より小さいサンプル日本語フォントが含まれています。使用方法については、/usr/dt/README の "Sample small fonts for JA_JP(Japanese UTF-8)" を参照してください。

AIX V4.3.3 以降でサポートされている X サーバーには、TrueType フォントを表示するラスタライザー機能が含まれています。また、AIX V5.1 の X サーバーから新しいラスタライザー機能が提供されます。このフォント・ラスタライザーは、より美しい TrueType フォントを提供し、OpenType フォントもサポートしています。

7.5 JA_JP ロケールの使用方法

JA_JP ロケールは、共通デスクトップ環境でのみ使用できます。

7.5.1 ログイン

共通デスクトップ環境にログインする時に、ログイン画面のオプション・ボタンから[言語]メニューを表示し、[日本語 UTF-8]を選択してください。環境変数 LANG は JA_JP に設定されます。

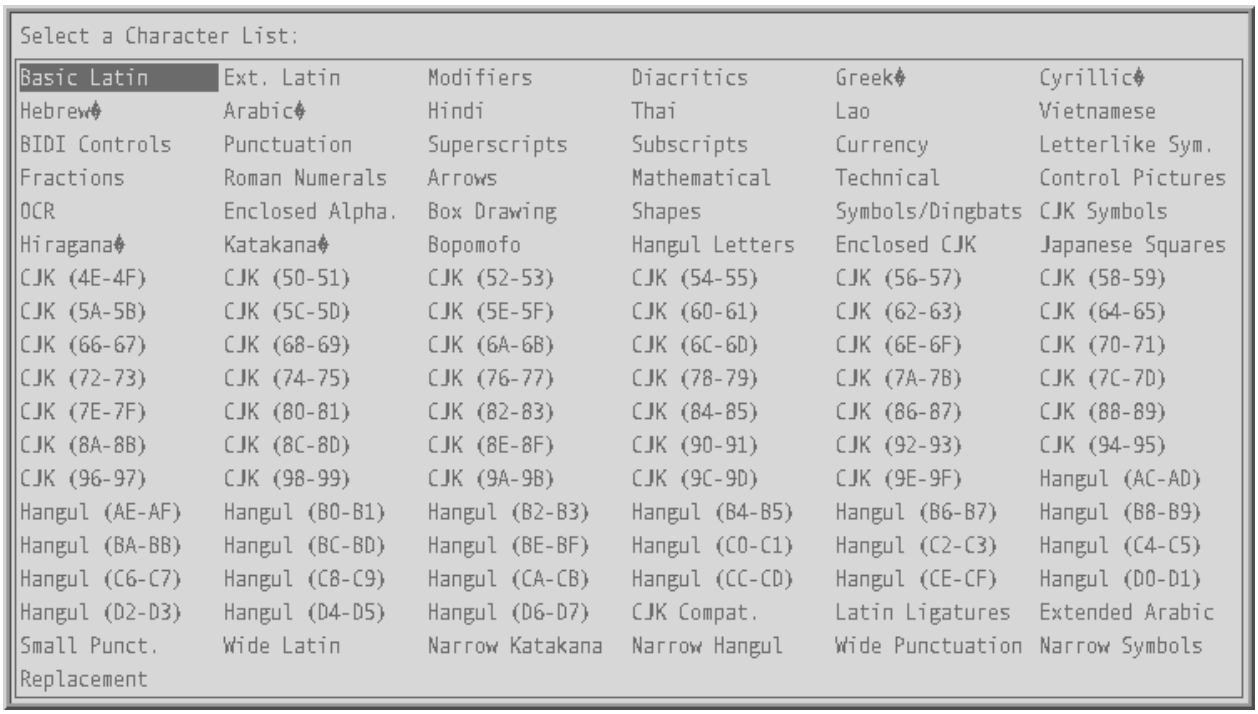
7.5.2 入力方法

すべての UTF-8 ロケールでは、入力機能として Universal Input Method が使用されています。Universal Input Method は、デフォルトとして各国語の Input Method を呼びだします。したがって、JA_JP ロケールで文字入力を行う際には、通常の日本語入力機能が使用できます。日本語入力機能で入力できない文字を入力する場合は、Universal Input Method は以下のマルチリンガル入力機能をサポートしています。

UTF-8 コードセットをサポートします。

入力機能(Input Method)の切り換えをサポートします。Ctrl キーと left Alt キーと文字 i を同時に押してください。下図のような Input Method の一覧メニューが表示されますので、マウス・ボタンで使用したい Input Method を選択してください。

ポイント&クリック入力をサポートします。Ctrlキーと　left Altキーと文字 l を同時に押してください。下図のような Unicode の文字カテゴリーの一覧メニューが表示されます。



メニューから任意のマウス・ボタンで文字カテゴリを選ぶと、文字の一覧表が表示されます。例えば、Basic Latin を選択すると下図のような表が表示されますので、任意のマウス・ボタンで使用したい文字をクリックしてください。

終了するには、Ctrlキー と left Altキー と 文字 c を同時に押してください。

7.6 外字領域

Unicode では、UCS-2 の 0xE000 から 0xF8FF までの領域が、PUA(Private Use Area)として定義されており、ユーザー定義可能領域として利用することができます。

日本語 Ja_JP ロケールの IBM-943 コードセットでは、0xF040 から 0xF9FC までの 1880 文字がユーザー定義文字の領域として確保されていますが、これらは、Unicode では以下のコードポイントに対応しています。

IBM-943	UCS-2	UTF-8
f040	e000	ee8080
:	:	:
f9fc	e757	ee9d97

AIX では、上記の 1880 文字が日本語ロケール共通のユーザー定義文字の領域として使用できます。

注）Unicode の PUA は各国語で独立したものではありません。
例えば、韓国語や中国語のユーザー定義文字も UCS-2 の 0xE000 から順に割り当てられます。

このページは空白です。

8. JISX0213 サポート

■ 目次

8.1 JISX0213 文字セット

ここでは、AIX Version 5.2 より新たにサポートされた JIS X 0213 文字セットについて紹介します。

8.1.1 JISX0213:2000 の概要

JISX0213とは従来のJISX0208を補うものとして制定された規格です。漢字を第3水準及び第4水準まで拡張し、これまで不足していた人名用、地名用、教育用の漢字・記号類等が追加収録されています。JISX0213に含まれる合計11223文字の内訳は以下の通りです。

JISX0208に含まれる文字	6879字
非漢字及び第3水準漢字	1908字
第4水準漢字	2436字

AIX V5.2より、JIS第3水準および第4水準を利用するユーザーのための64ビット・アプリケーションの開発環境および実行環境が、JA_JPロケールで提供されています。JISX0213文字セットのほとんどは、Unicodeでは16ビットのUCS-2に含まれますが、一部はExtension Bとして32ビットのUTF-32(UCS-4)の範囲に含まれます。AIX V5.3では、プロセス・コードとして32ビット環境ではUCS-2を、64ビット環境ではUTF-32をサポートしています。このため、AIX V5.3の64ビット環境でのみJISX0213文字セットの全文字を取り扱うことができます。

8.1.2 使用手順

JIS第3水準および第4水準の文字を入力可能にするには、日本語処理機能プロファイルにオプション "levelkj" を設定します。このオプションは、64ビット・アプリケーションの区点入力で入力可能な文字セットの範囲を設定します。日本語処理機能プロファイルの詳細は「1.1.6 日本語処理機能のカストマイズ」を参照してください。

levelkj : 0208	第1、2水準およびIBM拡張文、NEC選定文字、NECのIBM選定文字が 入力可能な従来のモード（省略時）
levelkj : 0213-3	第3水準まで入力可能なJISX0213モード
levelkj : 0213-4	第4水準まで入力可能なJISX0213モード

- すでに辞書に登録された語句を入力する場合や、後述する区点入力以外の入力方法はこのオプションに依存しません。

- levelkjに0213-3または0213-4が指定されたときには、次の順序でユーザー辞書を参照します。従来のユーザー辞書は使用されません。

    1) 環境変数JIMUSRDICTに指定されたもの
    2) $HOME/.usrdict0213
    3) /usr/lpp/jls/dict/usrdict0213

- 同一の辞書ファイルを 0208 モードと 0213-3 または 0213-4 モードとで共用しないでください。

8.1.3 入力方法

【JIS区点入力】
漢字番号キーを押してください。JIS区点入力用のウィンドウが表示されますので、JIS区点番号を入力してください。ただし、第2面の第n区は94+n区とし、2面1区2点ならば9502と入力します。（一つの面には94区が、一つの区には94点が含まれます。第2面は第1面からの連続とみなし、2面1区を95区と数えます。）

【コード入力】
CtrlキーとAltキーと文字uを同時に押してください。Unicode入力用のウィンドウが表示されますので、UCS-2（16ビット）またはUTF-32（32ビット）の16進（Hex）コードを入力してください。

【ポイント&クリック入力】
CtrlキーとAltキーと文字lを同時に押してください。Unicodeの文字カテゴリーの一覧メニューが表示されます。メニューから文字カテゴリを選ぶと文字の一覧表が表示されますので、使用したい文字をクリックしてください。

- コード入力、及びポイント&クリック入力では JISX0213 以外の文字セットも入力できますが、辞書に登録することはできません。

8.1.4 ユーザー辞書ユーティリティー

JISX0213用のユーザー辞書に語句を登録するには、専用のユーティリティーを使用します。JISX0213用のユーザー辞書ユーティリティーは、次のコマンドにより始動します。

```
kjdict0213 [-d 辞書名] [-y 読み -g 語句 | -l] [-f]
```

```
-d  登録・変更したいユーザー辞書ファイルを指定します。
-y  コマンド・ラインで読みを指定します。
-g  コマンド・ラインで語句を指定します。
-l  コマンド・ラインで一覧表を出力します。
-f  処理を強制的に開始します。
```

-dを指定しなかった場合、次の順序でユーザー辞書を参照します。

1) 環境変数JIMUSRDICTに指定されたもの
2) $HOME/.usrdict0213
3) /usr/lpp/jls/dict/usrdict0213

- kjdict0213はGUIアプリケーションですが、使用方法は従来のkjdictに沿ったものです。ただし「組合せ」と「回復」はサポートされません。これらの機能は、kjdictを使用してください。kjdictの詳細は「2. ユーザー辞書ユーティリティー」のを参照してください。

- 従来のユーザー辞書の内容をJISX0213用のユーザー辞書に移行するには、以下の方法で行うと便利です。

    1) kjdictで一覧表をファイルに出力します。
    2) 出力ファイルを編集して、登録したい読みと語句を以下のようなkjdict0213のコマンド・ライン形式に記述し直します。

```
kjdict0213 -y 読み -g 語句
kjdict0213 -y 読み -g 語句
    :
```

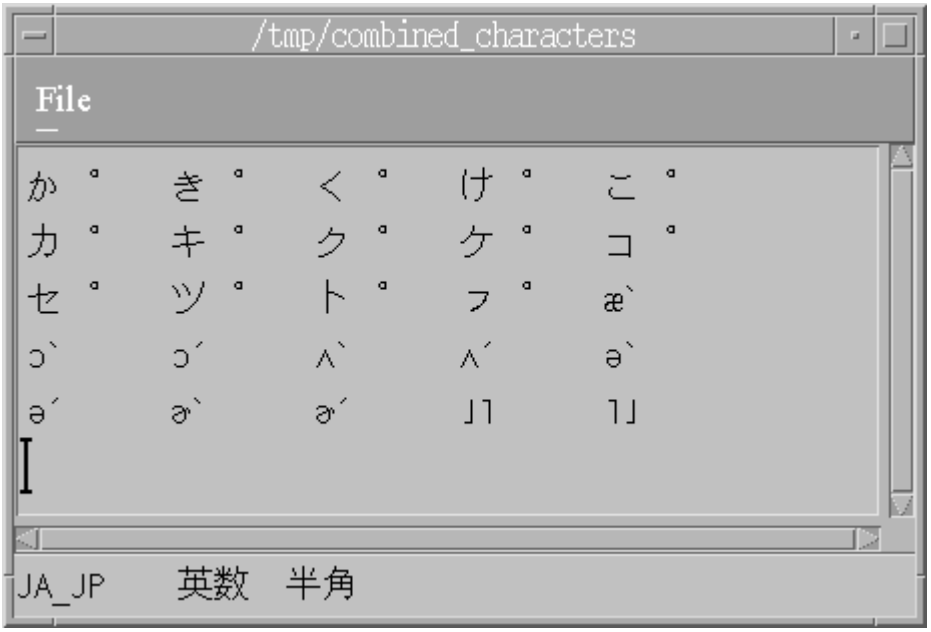
    3) 作成されたファイルをshell scriptとして実行します。

8.1.5 合成文字

JISX0213に含まれる25文字は該当する字形がUnicodeで定義されていないため、二つの文字の組み合わせで入力されます。以下にこれらの区点と入力されるUTF-8のコードポイント、およびUCS-2でのコードを示します。

区点	UTF-8コード	UCS-2コード
0487	E3818B+E3829A	304B+309A
0488	E3818D+E3829A	304D+309A
0489	E3818F+E3829A	304F+309A
0490	E38191+E3829A	3051+309A
0491	E38193+E3829A	3053+309A
0587	E382AB+E3829A	30AB+309A
0588	E382AD+E3829A	30AD+309A
0589	E382AF+E3829A	30AF+309A
0590	E382B1+E3829A	30B1+309A
0591	E382B3+E3829A	30B3+309A
0592	E382BB+E3829A	30BB+309A
0593	E38384+E3829A	30C4+309A
0594	E38388+E3829A	30C8+309A
0688	E387B7+E3829A	31F7+309A
1136	C3A6+CC80	00E6+0300
1140	C994+CC80	0254+0300
1141	C994+CC81	0254+0301
1142	CA8C+CC80	028C+0300
1143	CA8C+CC81	028C+0301
1144	C999+CC80	0259+0300
1145	C999+CC81	0259+0301
1146	C99A+CC80	025A+0300
1147	C99A+CC81	025A+0301
1169	CBA9+CBA5	02E9+02E5
1170	CBA5+CBA9	02E5+02E9

下図に64ビット・アプリケーションxmtexted64にて合成文字を入力した例を示します。



AIX complex text layout services の提供する ActiveShapeEditing 機能を利用することで、これら二つの文字を合成文字として表示することができます。付録「K. プログラム・リスト losample.c」にレイアウト関数を用いたサンプル・プログラムを示します。

レイアウト機能の詳細は「AIX バージョン6.1 ナショナル・ランゲージ・サポートガイドおよびリファレンス」(SC88-4576)の「レイアウト(両方向テキストと字形)の概要」を参照してください。

また、これらの合成文字は単一文字のフォントもUnicodeの外字領域に用意されています。以下にUTF-8のコードポイント、およびUCS-2でのコードを示します。

区点	UTF-8コード	UCS-2コード
0487	EF9CA0	F720
0488	EF9CA1	F721
0489	EF9CA2	F722
0490	EF9CA3	F723
0491	EF9CA4	F724
0587	EF9CA5	F725
0588	EF9CA6	F726
0589	EF9CA7	F727
0590	EF9CA8	F728
0591	EF9CA9	F729
0592	EF9CAA	F72A
0593	EF9CAB	F72B
0594	EF9CAC	F72C
0688	EF9CAD	F72D
1136	EF9CAE	F72E
1140	EF9CAF	F72F
1141	EF9CB0	F730
1142	EF9CB1	F731
1143	EF9CB2	F732
1144	EF9CB3	F733
1145	EF9CB4	F734
1146	EF9CB5	F735
1147	EF9CB6	F736
1169	EF9CB7	F737
1170	EF9CB8	F738

JIS区点入力で外字領域の方の文字を入力するには、日本語処理機能プロファイルにオプション ”puacomb” を設定します。このオプションは、入力される合成文字の種類を設定します。指定できるオプションの値は、以下のいずれかです。64ビット・アプリケーションの区点入力で、levelkjオプションにJISX0213モードが設定されている場合に有効です。

puacomb : off	二つの文字の組み合わせで入力(defaultの動作)
puacomb : on	外字領域の一つの文字を入力

- 外字領域の合成文字をユーザー辞書に登録することはできません。このため、ユーザー辞書ユーティリティーは常に二つの文字の組み合わせで入力します。また、外字領域の文字はJISX0213には含まれないため、ESC/Pデータ・ストリームでは、未定義な文字として<?>が印刷されます。

8.1.6 日本語プリンター

以下のデータ・ストリームでは、qprtコマンドでコードページをUTF-8に指定することにより、JISX0213の文字を印刷することができます。第3水準および第4水準の文字を印字する場合、その文字の登録してあるフォント・ファイルの名前を指定します。

データ・ストリーム	ファイルセット
ESC/P	printers.escpj84_JP.rte
PAGES	printers.ibm-pages_JP.rte
LIPS4	printers.lips4_JP.rte
LIPS2+, LIPS3	printers.canlbp-A404F_JP.rte printers.canlbp-B406G.rte printers.canlbp.rte

qprtコマンドで使用するフラグを説明します。

-X <コードページ>

<コードページ>	UTF-8を指定します。入力ファイルはUTFコード(JA_JP)として処理されます。

-F <Unicodeフォント・ファイル>,<Extension Bフォント・ファイル>

<Unicodeフォント・ファイル>	Unicodeのフォント・ファイル名を絶対パス名で指定します。
<Extension Bフォント・ファイル>	Unicode Extension Bのフォント・ファイル名を絶対パス名で指定します。

ただし、ESC/Pデータ・ストリームでは以下の二つのフラグでフォント・ファイルを指定します。

-I <フォント・パス>

<フォント・パス>	-Fで指定するフォント・ファイルのあるディレクトリーを指定します。

-F <Unicodeフォント・ファイル>,<Extension Bフォント・ファイル>

<Unicodeフォント・ファイル>	Unicodeのフォント・ファイルの名前を指定します。
<Extension Bフォント・ファイル>	Unicode Extension Bのフォント・ファイルの名前を指定します。

使用例）

```
$ qprt -Plips -XUTF-8 -F/tmp/smw24.pcf.Z,/tmp/smweb24.pcf.Z outfile

$ qprt -Pescp -XUTF-8 -I/tmp -Fsmw24.pcf.Z,smweb24.pcf.Z outfile
```

- qprtコマンドの各フラグの引数に指定できる文字列の長さは80バイトまでです。
- -Fフラグでは圧縮されたPCFファイル名を指定してください。ただし、AIXで提供されているJISX0213の文字が登

録されたフォントはTrueTypeのみのため、PCFファイルの作成は以下の例に示す手順で行います。

1) 以下の各ファイルセットを導入してください。

```
X11.fnt.fontServer
X11.fnt.util
X11.samples.fnt.util
```

2) フォント・サーバーを起動します。

```
xfs &
```

3) BDFファイルを作成します。Unicodeフォントの場合は:

```
fstobdf -s tcp/HOSTNAME:7100 -fn "-monotype-sansmonowt-medium-r-normal--24-173-100-100-m-221-ucs2.cjk_japan-0" >smw24.bdf
```

Unicode Extension Bフォントの場合は:

```
fstobdf -s tcp/HOSTNAME:7100 -fn "-monotype-sansmonowtextb-medium-r-normal--24-173-100-100-m-230-unicode-2" >smweb24.bdf
```

推奨されるピクセル・サイズは24です。ただし、使用するプリンターによって適切なフォントの大きさが異なる場合があります。

4) PCFファイルに変換します。

```
bdftopcf -o smw24.pcf smw24.bdf
bdftopcf -o smweb24.pcf smweb24.bdf
```

5) 圧縮します。

```
compress smw24.pcf
compress smweb24.pcf
```

以下にコードページUTF-8で印刷可能な文字の詳細を示します。

正しいフォント・ファイルが指定されている場合、すべての JISX0213 の文字を印刷することができます。NEC 選定文字の 1 字、IBM 拡張文字(NEC の IBM 選定文字)の 84 字は JISX0213 では定義されていません。これらの文字は ESC/P データ・ストリームでは、未定義な文字として<?>が印刷されます。

JISX0213 で未定義な NEC 選定文字と UTF コード

Σ E28891

JISX0213 で未定義な IBM 拡張文字(NEC の IBM 選定文字)と UTF コード

任	E4BBBC	但	E4BCB9	俔	E4BF8D	傀	E4BFBF
僩	E583B4	儆	E58398	糲	E585A4	冷	E586BE
凪	E587AC	刋	E58A9C	劾	E58B80	卲	E58DB2
圾	E58F9D	嫍	EFA88E	垔	E59D99	坦	E59DA5
墶	E5A2B2	夥	E5A593	奐	E5A5A3	妹	E5A6BA
峐	E5B3B5	嶄	E5B790	弭	E5BCA1	恕	E6819D
悅	E68285	愡	E6839E	愠	E684A0	愦	E68491
或	E68893	敎	E6958E	昂	E698BB	昜	E698AE
晴	EFA892	朐	E69C8E	檄	E6ABA2	汯	E6B1AF
浯	E6B5AF	淚	E6B696	清	E6B7B8	滬	E6B7B2
淘	E6B8B9	狹	E78CA4	珣	E78EBD	珒	E78F92
珵	E78FB5	琩	E790A9	皂	E79A82	益	EFA897
硢	E7A1BA	礼	EFA898	靖	EFA89C	精	EFA89D
羨	E7BEA1	羽	EFA89E	奉	E88FB6	董	E895AB
蠇	E8A087	讅	E8AD93	赶	E8B5B6	赴	EFA8A3
軏	E8BB8F	逸	EFA8A5	遧	E981A7	釦	E9879E
鉖	E98886	鉷	E989B7	鋂	EFA8A7	鋹	E98B95
鋅	EFA8A8	鋺	E98EA4	鏸	E98FB8	鐮	E990B1
鑼	E99188	閒	E99692	陽	EFA8A9	霬	E99D83
青	E99D91	飯	EFA8AA	飼	EFA8AB	餧	E9A4A7
館	EFA8AC	高	E9AB99	鮫	E9AEBB	鶴	EFA8AD

正しいフォント・ファイルが指定されていない場合、通常はプリンターがフォントを持っている JISX0208 の文字しか印刷されません(*1)。ただし、ESC/P データ・ストリームでは上記の未定義文字を除く NEC 選定文字が印刷できます(*2)。また、PAGES データ・ストリームでは NEC の IBM 選定文字と IBM 拡張文字が印刷できます(*2)。

*1 JISX0208 の文字として印刷される NEC 選定文字(9 字)、NEC の IBM 選定文字(1 字)と UTF コードを示します。

NEC 選定文字	≒ E28992	≡ E289A1	∫ E288AB
	√ E2889A	⊥ E28AA5	∠ E288A0
	∵ E288B5	∩ E288A9	∪ E288AA
NEC の IBM 選定文字	¬ EFBFA2		

*2 NEC 選定文字かつ IBM 拡張文字である 13 字と UTF コードを示します。

Ⅰ E285A0	Ⅱ E285A1	Ⅲ E285A2	Ⅳ E285A3	Ⅴ E285A4
Ⅵ E285A5	Ⅶ E285A6	Ⅷ E285A7	Ⅸ E285A8	Ⅹ E285A9
㈱ E388B1	№ E28496	℡ E284A1		

8.1.7 制限事項

- JA_JP(Unicode)ロケールの64ビット・アプリケーションを対象とし、Ja_JP(SJIS)、またはja_JP(EUC)ロケールや32ビット・アプリケーションはサポートされません。ただし、JA_JPロケールの32ビット・アプリケーションでは、JIS第3水準および第4水準の文字の内、UCS-2の値をもつ文字だけを取り扱う事ができます。

- 日本語プリンターへの出力はESC/P、PAGES、LIPS2+、LIPS3およびLIPS4のデータ・ストリームをサポートしています。ポストスクリプトには対応していません。

- JISX0213の文字を含むUTF-8ファイルの互換性について、以下の表にAIXの各バージョンでサポートされる文字の種類を示します。

	V5.1	V5.2	V5.3以降
JISX0208の文字	○	○	○
UCS-2コードのJISX0213文字	― (*1)	○	○
UTF-32コードのJISX0213文字	― (*2)	― (*2)	○ (*3)

*1　Unicodeのフォントに文字がある場合には表示されます。
*2　不正な文字として扱われるため、文字化けや切り捨てが発生します。
*3　64ビット環境でのみサポートされます。

8.2 JISX0213:2004

8.2.1 概要

AIX V5.3 ML 5300-05より、2004年に改正された「7ビット及び8ビットの2バイト情報交換用符号化拡張漢字集合(追補1) - JIS X0213:2004」がJA_JP(Unicode)ロケールの64ビット・アプリケーション開発環境にて提供されます。JISX0213:2004における主な改正内容を示します。

- 168字の例示字形の字体が変更されました。内訳は以下の通りです。付録に変更された文字の一覧表を示します。

  第1水準　139字
  第2水準　28字
  第3水準　1字

- JISX0213:2000 で未定義であった以下の第 3 水準の 10 個所に新たな 10 字が追加されました。文字数は合計で 11233 字となりました。

  区点(UCS) : 1401(4FF1)　1594(525D)　4752(20B9F)　4794(541E)　8407(5653)
  9490(59F8)　9491(5C5B)　9492(5E77)　9493(7626)　9494(7E6B)

JISX0213:2004の規格詳細については、日本工業標準調査会のホームページ(http://www.jisc.go.jp/)でJIS検索よりJIS規格番号「X0213」を検索することにより参照できます。

8.2.2 2004 年改正での人名用漢字

2004年に戸籍法が改正され人名用漢字が合計で983字に改正されました。内訳は以下の通りです。

改正前の人名用漢字	290字
改正前の人名用漢字許容字体	205字
新規追加	488字

経済産業省ホームページ(http://www.meti.go.jp/press/20041221008/20041221008.html)より「人名用漢字に対する JIS 漢字コード表の対応状況の公表について」を参照できます。

JISX0213:2004 はこの 2004 年改正での人名用漢字に対応しているため、JA_JP ロケール(UTF-8)の JISX0213 漢字コード表を用いれば、新しい人名用漢字をすべて使用することができます。

8.2.3 PC コードおよび EUC コードにおける改正人名用漢字への対応方法

AIX上のPCコード(Ja_JPロケール)およびEUCコード(ja_JPロケール)で使用できる日本語フォントのうち、以下のJISX0208.1983-0 漢字フォントの字体が変更になりました。変更された文字は、JISX0213:2004の168字の例示字形の字体変更のうちの第1水準139字、第2水準28字に対応しています。

- 標準体フォント

  Kanji06.pcf.Z
  Kanji09.pcf.Z
  k14.pcf.Z
  jiskan16.pcf.Z
  Kanji12S.pcf.Z
  Kanji12.pcf.Z
  Kanji17S.pcf.Z
  Kanji17.pcf.Z
  Kanji23S.pcf.Z
  Kanji23.pcf.Z
  Kanji12DS.pcf.Z
  Kanji12D.pcf.Z
  jiskan24.pcf.Z
  Kanji23GS.pcf.Z
  Kanji23G.pcf.Z

- 太字体フォント

  Kanji12B.pcf.Z
  Kanji17B.pcf.Z
  Kanji23B.pcf.Z
  Kanji23GB.pcf.Z

- 斜字体フォント

  Kanji12I.pcf.Z
  Kanji17I.pcf.Z
  Kanji23I.pcf.Z
  Kanji23GI.pcf.Z

また、これらの変更前(旧字体)のフォントは、以下のAIX Japan Kit V2.4のファイルセットにより提供されています。

ファイルセット	機能名
Jkit.fonts.compat	Japanese PCF Fonts Compatibility

上記ファイルセットの導入後に、互換フォント設定用コマンド pcfsetupを実行すると、旧字体のフォントが使用可能となります。

<pcfsetup コマンド>

[文法]
　　/usr/lpp/jkit/samples/bin/pcfsetup [-u]
[説明]
　　旧字体のJISX0208.1983日本語フォントを設定します。
　　-u フラグでこの設定を解除し、2004年改正の人名用漢字対応のフォントに戻します。
　　rootにてコマンドを実行した後、CDEにログインし直すか、Xサーバーを起動し直してください。

8.2.4 2004 年改正での人名用漢字対応字体への変更一覧表

注）JA_JPロケールで使用するUnicode TrueType FontsはOut-line GlyphのみがJISX0213:2004に対応しています。このためdtterm等で使用するフォントサイズ"25.5ポイント"(-dt-interface user-medium-r-normal-xxl*)以外のサイズでは旧字体となります。
　　フォントによっては旧字体が異なる場合があります。

JIS 区点	PC コード	EUC コード	UTF コード	旧字体	新字体
1609	88A7	B0A9	E980A2	逢	逢
1618	88B0	B0B2	E88AA6	芦	芦
1627	88B9	B0BB	E9A3B4	飴	飴
1678	88EC	B0EE	E6BAA2	溢	溢
1681	88EF	B0F1	E88CA8	茨	茨
1683	88F1	B0F3	E9B0AF	鰯	鰯
1692	88FA	B0FC	E6B7AB	淫	淫
1710	8949	B1AA	E8BF82	迂	迂
1725	8958	B1B9	E58EA9	厩	厩
1729	895C	B1BD	E59982	噂	噂
1734	8961	B1C2	E9A48C	餌	餌
1808	89A6	B2A8	E8A596	襖	襖
1864	89DE	B2E0	E8BFA6	迦	迦
1871	89E5	B2E7	E78999	牙	牙
1886	89F4	B2F6	E5BBBB	廻	廻
1890	89F8	B2FA	E681A2	恢	恢
1902	8A41	B3A2	E699A6	晦	晦
1910	8A49	B3AA	E89FB9	蟹	蟹
1975	8A8B	B3EB	E8919B	葛	葛
1983	8A93	B3F3	E99E84	鞄	鞄
1988	8A98	B3F8	E9879C	釜	釜
2045	8ACB	B4CD	E7BFB0	翰	翰
2069	8AE3	B4E5	E7BFAB	翫	翫
2111	8B4A	B5AB	E5BEBD	徽	徽
2132	8B5F	B5C0	E7A587	祇	祇
2166	8B82	B5E2	E6B1B2	汲	汲
2168	8B84	B5E4	E781B8	灸	灸
2172	8B88	B5E8	E7AC88	笈	笈
2210	8BA8	B6AA	E58DBF	卿	卿
2234	8BC0	B6C2	E9A597	饗	饗

JIS 区点	PC コード	EUC コード	UTF コード	旧字体	新字体
2247	8BCD	B6CF	E58385	僅	僅
2284	8BF2	B6F4	E596B0	喰	喰
2291	8BF9	B6FB	E6AB9B	櫛	櫛
2293	8BFB	B6FD	E5B191	屑	屑
2309	8C48	B7A9	E7B282	粂	粂
2323	8C56	B7B7	E7A581	祁	祁
2368	8C84	B7E4	E99A99	隙	隙
2381	8C91	B7F1	E580A6	倦	倦
2394	8C9E	B7FE	E68DB2	捲	捲
2403	8CA1	B8A3	E789BD	牽	牽
2416	8CAE	B8B0	E98DB5	鍵	鍵
2433	8CBF	B8C1	E8ABBA	諺	諺
2511	8D4A	B9AB	E5B7B7	巷	巷
2528	8D5B	B9BC	E6A297	梗	梗
2549	8D70	B9D1	E8868F	膏	膏
2584	8D94	B9F4	E9B5A0	鵠	鵠
2589	8D99	B9F9	E79491	甑	甑
2621	8DB3	BAB5	E58F89	叉	叉
2671	8DE5	BAE7	E6A68A	榊	榊
2707	8E46	BBA7	E896A9	薩	薩
2710	8E49	BBAA	E9AF96	鯖	鯖
2712	8E4B	BBAC	E98C86	錆	錆
2713	8E4C	BBAD	E9AEAB	鮫	鮫
2733	8E60	BBC1	E9A490	餐	餐
2861	8EDB	BCDD	E69D93	杓	杓
2862	8EDC	BCDE	E781BC	灼	灼
2922	8F55	BDB6	E9858B	酋	酋
2961	8F7C	BDDD	E6A5AF	楯	楯
2982	8F92	BDF2	E896AF	薯	薯
2983	8F93	BDF3	E897B7	藷	藷

JIS 区点	PC コード	EUC コード	UTF コード	旧字体	新字体
3005	8FA3	BEA5	E593A8	哨	哨
3068	8FE2	BEE4	E99E98	鞘	鞘
3083	8FF1	BEF3	E69D96	杖	杖
3110	9049	BFAA	E89D95	蝕	蝕
3154	9075	BFD6	E8A88A	訊	訊
3164	9080	BFE0	E98097	逗	逗
3202	90A0	C0A2	E691BA	摺	摺
3281	90EF	C0F1	E692B0	撰	撰
3289	90F7	C0F9	E7858E	煎	煎
3290	90F8	C0FA	E785BD	煽	煽
3292	90FA	C0FC	E7A9BF	穿	穿
3293	90FB	C0FD	E7AEAD	箭	箭
3307	9146	C1A7	E8A9AE	詮	詮
3325	9158	C1B9	E5998C	噌	噌
3344	916B	C1CC	E981A1	遡	遡
3423	91B5	C2B7	E68F83	揃	揃
3429	91BB	C2BD	E9819C	遜	遜
3460	91DA	C2DC	E885BF	腿	腿
3493	91FB	C2FD	E89BB8	蛸	蛸
3509	9248	C3A9	E8BEBF	辿	辿
3514	924D	C3AE	E6A8BD	樽	樽
3523	9256	C3B7	E6AD8E	歎	歎
3580	9290	C3F0	E8A8BB	註	註
3585	9295	C3F5	E780A6	瀦	瀦
3629	92BB	C4BD	E68D97	捗	捗
3640	92C6	C4C8	E6A78C	槌	槌
3642	92C8	C4CA	E98E9A	鎚	鎚
3652	92D2	C4D4	E8BEBB	辻	辻
3682	92F0	C4F2	E68CBA	挺	挺
3702	9341	C5A2	E984AD	鄭	鄭

JIS 区点	PC コード	EUC コード	UTF コード	旧字体	新字体
3707	9346	C5A7	E693A2	擢	擢
3714	934D	C5AE	E6BABA	溺	溺
3738	9365	C5C6	E5858E	兎	兎
3740	9367	C5C8	E5A0B5	堵	堵
3743	936A	C5CB	E5B1A0	屠	屠
3750	9371	C5D2	E8B3AD	賭	賭
3852	93D2	C6D4	E7809E	瀞	瀞
3859	93D9	C6DB	E98181	遁	遁
3870	93E4	C6E6	E8AC8E	謎	謎
3871	93E5	C6E7	E78198	灘	灘
3874	93E8	C6EA	E6A5A2	楢	楢
3909	9448	C7A9	E7A6B0	禰	禰
3955	9476	C7D7	E7898C	牌	牌
3971	9487	C7E7	E98099	這	這
3973	9489	C7E9	E7A7A4	秤	秤
3993	949D	C7FD	E9A781	駁	駁
4004	94A2	C8A4	E7AEB8	箸	箸
4032	94BE	C8C0	E58F9B	叛	叛
4052	94D2	C8D4	E68CBD	挽	挽
4080	94EE	C8F0	E8AAB9	誹	誹
4085	94F3	C8F5	E6A88B	樋	樋
4103	9542	C9A3	E7A897	稗	稗
4115	954E	C9AF	E980BC	逼	逼
4121	9554	C9B5	E8ACAC	謬	謬
4131	955E	C9BF	E8B1B9	豹	豹
4132	955F	C9C0	E5BB9F	廟	廟
4146	956D	C9CE	E78095	瀕	瀕
4164	9580	C9E0	E696A7	斧	斧
4235	95C1	CAC3	E894BD	蔽	蔽
4245	95CB	CACD	E79EA5	瞥	瞥

JIS 区点	PC コード	EUC コード	UTF コード	旧字体	新字体
4246	95CC	CACE	E89491	蔑	蔑
4251	95D1	CAD3	E7AF87	篇	篇
4258	95D8	CADA	E5A8A9	娩	娩
4260	95DA	CADC	E99EAD	鞭	鞭
4289	95F7	CAF9	E5BA96	庖	庖
4309	9648	CBA9	E893AC	蓬	蓬
4380	9690	CBF0	E9B192	鱒	鱒
4388	9698	CBF8	E8BF84	迄	迄
4457	96D7	CCD9	E584B2	儲	儲
4463	96DD	CCDF	E9A485	餅	餅
4466	96E0	CCE2	E7B1BE	籾	籾
4476	96EA	CCEC	E788BA	爺	爺
4490	96F8	CCFA	E99193	鑓	鑓
4492	96FA	CCFC	E68488	愈	愈
4518	9751	CDB2	E78CB7	猷	猷
4690	97F8	CEFA	E6BCA3	漣	漣
4691	97F9	CEFB	E78589	煉	煉
4692	97FA	CEFC	E7B0BE	簾	簾
4717	9850	CFB1	E6A694	榔	榔

JIS 区点	PC コード	EUC コード	UTF コード	旧字体	新字体
4945	996C	D1CD	E586A4	冤	冤
5055	99D5	D2D7	E58F9F	叟	叟
5091	99F9	D2FB	E592AC	咬	咬
5162	9A7D	D3DE	E598B2	嘲	嘲
5183	9A93	D3F3	E59B80	囀	囀
5549	9C70	D7D1	E5BE98	徘	徘
5708	9D47	D9A8	E68981	扁	扁
5989	9E99	DBF9	E6A398	棘	棘
6084	9EF2	DCF4	E6A999	橙	橙
6436	E0C2	E0C4	E78BA1	狡	狡
6517	E150	E1B1	E79495	甕	甕
6520	E153	E1B4	E794A6	甦	甦
6554	E175	E1D6	E796BC	疼	疼
6714	E24D	E3AE	E7A59F	祟	祟
6762	E27D	E3DE	E7AB88	竈	竈
6807	E2A5	E4A7	E7ADB5	筵	筵
6832	E2BE	E4C0	E7AF9D	籌	籌
7107	E446	E7A7	E885B1	腱	腱
7159	E47A	E7DB	E88998	艘	艘
7174	E48A	E7EA	E88A92	芒	芒
7342	E569	E9CA	E89994	虔	虔
7371	E587	E9E7	E89C83	蜜	蜜
7404	E5A2	EAA4	E8A085	蠅	蠅
7535	E662	EBC3	E8A89D	訝	訝
8043	E8C9	F0CB	E99D84	鬻	鬻
8055	E8D5	F0D7	E99DB1	靱	靱
8157	E978	F1D9	E9A899	騙	騙
8277	E9EB	F2ED	E9B489	鴉	鴉

ＪＩＳ 区 点	Ｐ Ｃ コード	ＥＵＣ コード	ＵＴＦ コード	旧字体	新字体
4767	--	--	F0A19AB4		妛

8.3 文字検査ツール (Character Level Checker)

8.3.1 kjlck コマンド

kjlckコマンドは、ファイルに含まれる文字の種類を検査し、詳細な分析結果を表示します。また、特定の文字種別を対象にした削除や置換処理を行います。kjlckコマンドは、64ビット・アプリケーションです。32ビット・アプリケーション環境ではサポートされません。

【構文】

```
kjlck [-c コードセット名] -l|-a|-w|-i|-n|-j|-e|-d|-r|-h [-UB34NI] [-S 置換キャラクター]
      [-v|-V] [-t] ファイル名
```

【フラグ】

-c コードセット名
入力ファイルのコードセットを指定します。指定可能なコードセット名は以下の通りです。省略時は現在のロケールとなります。

```
SJIS    Ja_JP IBM-943
EUC     ja_JP IBM-eucJP
Unicode JA_JP UTF-8
```

-l 入力ファイルに含まれる文字・漢字のレベルを出力します。表示内容は以下の通りです。制御コードには1バイトの0x00～1F、0x7Fが含まれます。

```
入力ファイル名
ファイル内の文字数、行数、総バイト数
以下の分類による対象文字の文字数、総バイト数
X0201         半角のJISローマ字/カタカナ
X0208 non     JISX0208非漢字
Level 1       第1水準漢字
Level 2       第2水準漢字
IBM ext       IBM拡張文字(SJIS/EUCのみ)
NEC sel       NEC選定文字(SJISのみ)
NEC-IBM sel   NECのIBM選定文字(SJISのみ)
X0213 non     JISX0213で追加された非漢字(Unicodeのみ)
Level 3       第3水準漢字でUCS-2のもの(Unicodeのみ)
Level 3 extB  第3水準漢字でExt.Bのもの(Unicodeのみ)
Level 4       第4水準漢字でUCS-2のもの(Unicodeのみ)
Level 4 extB  第4水準漢字でExt.Bのもの(Unicodeのみ)
User def      外字領域(Unicodeの場合はPUA)
Other ucs2    上記以外のUTFのコードでUCS-2のもの(Unicodeのみ)
Other extB    上記以外のUTFのコードでExt.Bのもの(Unicodeのみ)
Control       制御コード(バイト数のみ)
Undefined     コードセット上で未定義のコード(バイト数のみ)
```

-a　Unicodeの場合、UTF-8上のマッピングが異なる文字をAIXからWindowsへ置換え、結果を出力します。

-w　Unicodeの場合、UTF-8上のマッピングが異なる文字をWindowsからAIXへ置換え、結果を出力します。

以下の文字が置換えられます。

	AIXのUTF-8	WindowsでのUTF-8
―	E28094<EM_DASH>	E28095<HORIZONTAL_BAR>
～	E3809C<WAVE_DASH>	EFBD9E<FULLWIDTH_TILDE>
‖	E28096<DOUBLE_VERTICAL_LINE>	E288A5<PARALLEL_TO>
−	E28892<MINUS_SIGN>	EFBC8D<FULLWIDTH_HYPHEN-MINUS>
¦	C2A6<BROKEN_BAR>	EFBFA4<FULLWIDTH_BROKEN_BAR>

-i　SJISの場合、重複文字をAIXの優先順位で置換え、結果を出力します。
JISX0208 << IBM拡張文字 << NEC選定文字 の順に置換えられます。

-n　SJISの場合、重複文字をWindowsの優先順位で置換え、結果を出力します。
JISX0208 << NEC選定文字 << IBM拡張文字 の順に置換えられます。

以下の文字が置換えられます。

	JISX0208	IBM拡張文字	NEC選定文字
Ⅰ		FA4A	8754
Ⅱ		FA4B	8755
Ⅲ		FA4C	8756
Ⅳ		FA4D	8757
Ⅴ		FA4E	8758
Ⅵ		FA4F	8759
Ⅶ		FA50	875A
Ⅷ		FA51	875B
Ⅸ		FA52	875C
Ⅹ		FA53	875D
№		FA59	8782
℡		FA5A	8784
㈱		FA58	878A
≒	81E0		8790
≡	81DF		8791
∫	81E7		8792
√	81E3		8795
⊥	81DB		8796
∠	81DA		8797
∵	81E6	FA5B *	879A
∩	81BF		879B
∪	81BE		879C
￢	81CA	FA54 *	

* FA54、FA5B はコードセットIBM-932における旧IBM拡張文字です。

-j　EUCの場合、13区のNEC選定文字にある重複文字を、JISX0208の同一文字、あるいは、IBM拡張文字の同一文字に置換えます。

-e　EUCの場合、13区のNEC選定文字にある重複文字に、JISX0208の同一文字、および、IBM拡張文字の同一文字から置換えます。

以下の文字が置換えられます。

	JISX0208	IBM拡張文字	13区のEUC
Ⅰ		8FF3AB	ADB5
Ⅱ		8FF3AC	ADB6
Ⅲ		8FF3AD	ADB7
Ⅳ		8FF3AE	ADB8
Ⅴ		8FF3AF	ADB9
Ⅵ		8FF3B0	ADBA
Ⅶ		8FF3B1	ADBB
Ⅷ		8FF3B2	ADBC
Ⅸ		8FF3B3	ADBD
Ⅹ		8FF3B4	ADBE
№.		8FF3B8	ADE2
℡		8FF3B9	ADE4
㈱		8FF3B7	ADEA
≒	A2E2		ADF0
≡	A2E1		ADF1
∫	A2E9		ADF2
√	A2E5		ADF5
⊥	A2DD		ADF6
∠	A2DC		ADF7
∵	A2E8		ADFA
∩	A2C1		ADFB
∪	A2C0		ADFC

-d　処理対象のコードポイントを削除し、結果を出力します。

-r　処理対象のコードポイントを置換キャラクターに変換し、結果を出力します。

-h　コマンドの構文と説明をstderrに出力します。

l、a、w、i、n、j、e、d、r、hのいずれかのフラグが指定可能です。

-v　未定義のコードの詳細情報を以下の形式でstderrに出力します。

ファイル内バイトカウント、行カウント、行内バイトカウント、16進コード

-V 未定義のコードの詳細情報を以下の形式でstderrに出力します。コードの詳細な分類も16進コードの下に示します。v、Vのいずれかが指定可能です。

```
ファイル内バイトカウント、行カウント、行内バイトカウント、16進コード
                                                    分類..
```

分類の表示内容は以下の通りです。

```
SJISの場合
S1          1バイトの未定義
Sa          2バイトの未定義、2バイト目が不正(JIS 1-94点外)
Sb          2バイトの未定義、JIS 1-94区内で未定義のコード
Sc          2バイトの未定義、JIS 115-120区内で未定義のコード
S2          その他の2バイトの未定義
EUCの場合
E1          1バイトの未定義
Ea          2バイトの未定義、2バイト目が不正(JIS 1-94点外)
Eb          2バイトの未定義、JIS 1-94区内で未定義のコード
Ec          2バイトの未定義、JISX0201カタカナ以外
Ed          2バイトの未定義、2バイト目が不正(JIS区外)
E2          その他の2バイトの未定義
EA          3バイトの未定義、3バイト目が不正(JIS 1-94点外)
EB          3バイトの未定義、IBM拡張文字以外のJISX0212のコード
EC          3バイトの未定義、IBM拡張文字でもJISX0212でも外字でもない
E3          その他の3バイトの未定義
Unicodeの場合
U1          UTF-8 1バイトの未定義
U2          UTF-8 2バイトの未定義
U3          UTF-8 3バイトの未定義
U4          UTF-8 4バイトの未定義
```

-t マルチバイト文字の途中でLFまたはEOFになっている可能性がある場合(Truncated文字となっている場合)、途切れた部分までの詳細情報を以下の形式でstderrに出力します。

```
TRUNC                    行カウント、行内バイトカウント、16進コード
```

以下のフラグは、dおよび rフラグで有効です。

-U AIX上のコードセットで未定義のコードポイントを処理します。

-B Unicodeの場合、Ext.Bの文字のコードポイントを処理します。

-3 Unicodeの場合、JISX0213で追加された非漢字と第3水準漢字のコードポイントを処理します。

-4 Unicodeの場合、JISX0213の第4水準漢字のコードポイントを処理します。

-N SJISの場合、NEC選定文字のコードポイントを処理します。

-I SJISの場合、NECのIBM選定文字のコードポイントを処理します。

-S 置換キャラクター
置換キャラクターを指定します。省略時は全角空白となります。

注) -cで指定するコードセット名が現在のロケールと同一でない場合、置換キャラクターには半角のJISローマ字を指定してください。

- 以下に使用例を示します。

1. SJISファイルSample1に含まれる文字種別を検査し、結果を表示します。
```
kjlck -c Ja_JP -l Sample1
```

2. UnicodeファイルSample2に含まれる未定義のコードを削除し、結果をResult2に出力します。また、Sample2における未定義コードの詳細情報と分類、および、Truncated文字の情報をDetail2に出力します。
```
kjlck -c UTF-8 -r -U -V -t Sample2 1> Result2 2> Detail2
```

以下のDetail2の表示例では、2行5バイト目(ファイルの先頭から13バイト目)から3バイトの未定義コードe2bbb9と1バイトの未定義コードe9が含まれていることを示しています。また、このe9はTruncated文字です。
```
        13,       2,       5,   0xe2bbb9e9
                                    U3    U1
TRUNC             2,       8,   0xe9
```

このページは空白です。

付録

■ 目次

A. コンパイラーの日本語サポート

日本語をサポートするコンパイラーを以下に列挙します。日本語を含むソースコードをコンパイルする場合には日本語処理オプションを必ず指定して下さい。

1. IBM C and C++ Compilers
2. VisualAge Compiler C++
3. XL Fortran Compiler

各コンパイラーの日本語処理オプションは次のフラグを指定します。

-qdbcs

B. 日本語 aixterm に対する制御コード

日本語ターミナル・エミュレーター（aixterm）がサポートする制御コードについて説明します。

1バイト制御コード

BEL　0x07（ベル）

BS　0x08
カーソルを 1 カラム左に移動します。但しカーソルがすでに画面の左端にある場合は移動しません。

HT　0x09
カーソルを次のタブ・ストップに移動します。但しカーソルがすでに画面の右端にある場合は移動しません。

LF　0x0A
LNM モードがセットされていない場合は、カーソルはひとつ下の行に移動します。LNM モードがセットされている場合は（デフォルト）カーソルは次の行の先頭に移動します。カーソルがすでに最下行にある時は、どちらの場合でも画面が一行分スクロール・アップします。この時、画面最上の一行は消え、最下端に新しい一行が挿入されます。

VT　0x0B
LF と同じ扱いです。

FF　0x0c
LF と同じ扱いです。

CR　0x0D
カーソルを次の行の先頭に移動します。カーソルがすでに最下行にある時は、どちらの場合でも画面が一行分スクロール・アップします。この時、画面最上端の一行は消え、最下端に新しい一行が挿入されます。

ESC　0x1B
複数バイト制御コードの始まりを示します。

0x00 から 0x06、0x0E から 0x1A 及び 0x1C から 0x1F のコードに対しては何の機能も割り当てられていません。すなわち、aixterm にこれらのコードが送られても無視されます。

複数バイト制御コード

CPL　ESC [数値 F　Cursor Preceding Line
カーソルを指定された行数だけ上に移動します。カーソルは行の先頭に移動します。カーソルがすでに画面の最上端の行にあり、HFWRAP モードがセットされている場合は、カーソルは最下端の行にラップします。HFWRAP がセットされていない場合は最上端の行の先頭に位置付けられます。

CPR　ESC [数値：数値 R　Cursor Position Report
現在のカーソルの位置を、適用業務プログラムに知らせます。1 番目の数値は行番号で、2 番目の数値はカラム番号です。但し、このデータ・ストリームが aixterm に送られた場合は、CUP と同様に取り扱われます。

CUB　ESC [数値 D　Cursor Backward
カーソルを指定された数値分だけ左に移動します。カーソルが行の先頭に達した場合は HFWRAP がセットされていれば、カーソルは上の行にラップします。画面最上端の行の場合は最下端の行にラップします。HFWRAP がセットされていなければ、カーソルは行の先頭で止まります。

CUD　　ESC [数値 B　　Cursor Down
カーソルを指定された数値分だけ下に移動します。カーソルが最下端の行に達した場合は HFWRAP がセットされていれば、カーソルは最上端の行にラップします。HFWRAP がセットされていない場合は、カーソルは最下端の行で止まります。

CUF　　ESC [数値 C　　Cursor Forward
カーソルを指定された数値分だけ右に移動します。カーソルが行の右端に達した場合は HFWRAP がセットされていれば、カーソルは 1 つ下の行にラップします。HFWRAP がセットされていない場合は、カーソルはその行の右端で止まります。

CUP　　ESC [数値 H　　Cursor Position
カーソルを指定された位置に移動します。1 番目の行番号を、2 番目の数値はカラム番号を表します。

CUU　　ESC [数値 A　　Cursor Up
カーソルを指定された数値分だけ上に移動します。カーソルが最上端の行に達した場合は HFWRAP がセットされていれば、カーソルは最下端の行にラップします。HFWRAP がセットされていない場合は、カーソルは最上端の行で止まります。

CUD　　ESC [数値 B　　Cursor Down
カーソルを指定された数値分だけ下に移動します。カーソルが最下端の行に達した場合は HFWRAP がセットされていれば、カーソルは最上端の行にラップします。HFWRAP がセットされていない場合は、カーソルは最下端の行で止まります。

DCH　　ESC [数値 P　　Delete Character
現在のカーソル位置から、その位置の文字も含めて数値で指定された文字数分だけ文字を消去します。消去された文字列より右側にあった文字列はカーソル位置まで移動し、その右側はクリアーされます。

DL　ESC [数値 M　　Delete Line
現在カーソルのある行から、その行も含めて数値で指定された行数分だけ行を消去します。消去された行より下にあった行はスクロールアップされ、画面の下側から空白が挿入されます。

EA　ESC [数値 O　　Erase Area
EL と同じです。

ECH　　ESC [数値 X　　Erase Character
カーソル位置から、その位置の文字も含めて数値で示された文字分だけクリアーします。

ED　ESC [数値 J　　Erase Display
数値が 0 の場合、現在のカーソル位置からその文字も含めて、画面の終端までの全ての文字をクリアーします。数値が 1 の場合、画面の先頭から現在のカーソル位置までカーソル位置の文字も含めて全ての文字をクリアーします。数値が 2 の場合、画面全体をクリアーします。

EL　ESC [数値 K　　Erase Line
数値が 0 の場合、カーソル位置の文字とそこから行の右側までの文字をクリアーします。数値が 1 の場合、カーソル位置の文字とその行の先頭からカーソル位置までの文字をクリアーします。数値が 2 の場合、カーソルのある行全体をクリアーします。

ICH ESC [数値@　　Insert Character
数値で示された分だけ空白文字を現在のカーソル位置の前に挿入します。カーソル位置は変化しません。

IL　ESC [数値 L　　Insert Line

数値で示された分だけ空白行を現在のカーソル位置の前に挿入します。カーソル位置にあった行及びそれより下にあった行は、数値で示された分だけスクロールダウンされます。カーソル位置は変化しません。

NEL　　ESC [数値 E　　Next Line

カーソルを次の行の先頭に移動します。カーソルがすでに画面の最下端の行にある場合は、1 行スクロールアップします。

RCP　　ESC [u　　Restore Cursor Position

SCP によってセーブされた位置にカーソル移動します。

RI　ESC L　　Reverse Index

カーソルを 1 行上に移動します。カーソルがすでに画面の最上行にある場合は、HFWRAP がセットされていればカーソルは最下端の行にラップします。HFWRAP がセットされていなければ、カーソルは移動しません。カーソルのカラム位置は変化しません。

RM ESC [数値 1　　Reset Mode

数値で指定されたモードをリセットします。数値をセミコロンで区切ることにより、同時に複数のモードを指定することができます。

数値	説明
20	LNM (Line Feed New Line モード)
4	IRM (Insert モード)

SCPESC [s　　Save Cursor Position

現在のカーソル位置をセーブします。以前にセーブされていた値は失われます。

SGR　　ESC [数値 m　　Set Graphic Rendition

画面の属性を設定します。

数値	説明
0	ノーマル
4	下線
7	リバース
8	インビジブル

SM ESC [数値 h　　Set Mode

数値で指定されたモードをセットします。数値をセミコロンで区切ることにより、同時に複数のモードを指定することができます。

数値	説明
20	LNM (Line Feed-New Line モード)
4	IRM (Insert モード)

SU　ESC [数値 S　　Scroll Up

画面全体を指定された数値分だけスクロールアップします。画面の下端から空白行が挿入されます。

C. AIX 日本語処理機能の制限事項

本節では、AIX 日本語処理機能を使用する上での制限事項や注意事項について解説します。

1. 日本語キーボードのキートップ上には、チルダ ”～”（”わ”のキー）及びバックスラッシュ ”＼”（”ろ”のキー）の文字が書かれているところがあります。これらの文字 ”～” や ”＼”は ASCII コードセットに含まれているもので、JIS コードセット（JISX0201 で規定されている JIS8 ビット文字コード）では ”‾” 及び ”¥” に対応しています。従って AIX 日本語処理機能のもとで ”～” や ”＼” のキーを押した場合には、代わりに ”‾” 及び ”¥” の文字が表示されます。ただし、JA_JP ロケールでは ”¥” の代わりに ”＼” が、”‾” の代わりに ”～” が表示されます。これは、UTF-8 コードセットではコードポイント 0x5C 及び 0x7E が、それぞれ ”＼” 及び ”～” に対応しているためです。

   AIX V5.2 以降の JA_JP ロケールでは、キートップ上の ”¥” 、”＼”、”‾”（”へ”のキー）及び ”～” にそれぞれ対応した文字が入力されます。従来と同様な入力（”¥” のキーでコードポイント 0x5C、”‾” のキーでコードポイント 0x7E）をおこなうには、keyboard ファイルを下記のようにカスタマイズします。

   例）/usr/lpp/X11/defaults/xmodmap/JA_JP@alt/keyboard

```
    keycode 21 = asciicircum     overline        NoSymbol
    keycode 22 = yen             bar             NoSymbol
```

   修正例…

```
    keycode 21 = asciicircum     asciitilde      NoSymbol
    keycode 22 = backslash       bar             NoSymbol
```

2. 日本語環境で英語環境用に作成されたプログラムを実行する場合、文字コードセットが異なる為文字化けが起こる場合があります。そのようなプログラムは適切な言語環境を設定して実行する必要があります。またキーボードのマッピングが日本語キーボードに対応していない場合は、入力して不具合が生じる可能性があります。

3. システム名、ユーザー名及びグループ名に日本語を使用することはできません。

D. JIS8 ビット文字コード表

ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p	未定義		未定義	―	タ	ミ		
1	SOH	DC1	!	1	A	Q	a	q			。	ア	チ	ム		
2	STX	DC2	〃	2	B	R	b	r			「	イ	ツ	メ		
3	ETX	DC3	#	3	C	S	c	s			」	ウ	テ	モ		
4	EOT	DC4	$	4	D	T	d	t			、	エ	ト	ヤ		
5	ENQ	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	BEL	ETB	´	7	G	W	g	w			ア	キ	ヌ	ラ		
8	BS	CAN	(	8	H	X	h	x			イ	ク	ネ	リ		
9	HT	EM	)	9	I	Y	i	y			ウ	ケ	ノ	ル		
A	LF	SUB	*	:	J	Z	j	z			エ	コ	ハ	レ		
B	VT	ESC	+	;	K	[	k	{			オ	サ	ヒ	ロ		
C	FF	FS	,	<	L	¥	l	|			ヤ	シ	フ	ワ		
D	CR	GS	―	=	M	]	m	}			ユ	ス	ヘ	ン		未定義
E	SO	RS	・	>	N	^	n	‾			ヨ	セ	ホ	゛		未定義
F	SI	US	／	?	O	_	o	DEL			ツ	ソ	マ	゜		未定義

2バイト・コード文字の1バイト目

2バイト・コード文字の1バイト目

注）日本語ターミナル・エミュレーターでは、コード 0x7c は“｜”と表示されます。

E. ローマ字入力ーかな文字対応表

ローマーかな対応表（1）

ローマ字					ひらがな					カタカナ				
a	i	u	e	o	あ	い	う	え	お	ア	イ	ウ	エ	オ
ka	ki	ku	ke	ko	か	き	く	け	こ	カ	キ	ク	ケ	コ
		cu		co	か		く		こ	カ		ク		コ
		qu					く					ク		
sa	si	su	se	so	さ	し	す	せ	そ	サ	シ	ス	セ	ソ
	shi					し					シ			
	ci		ce			し		せ			シ		セ	
ta	ti	tu	te	to	た	ち	つ	て	と	タ	チ	ツ	テ	ト
	chi	tsu				ち	つ				チ	ツ		
na	ni	nu	ne	no	な	に	ぬ	ね	の	ナ	ニ	ヌ	ネ	ノ
ha	hi	hu	he	ho	は	ひ	ふ	へ	ほ	ハ	ヒ	フ	ヘ	ホ
		fu					ふ					フ		
ma	mi	mu	me	mo	ま	み	む	め	も	マ	ミ	ム	メ	モ
ya	yi	yu	ye	yo	や	い	ゆ	いぇ	よ	ヤ	イ	ユ	イェ	ヨ
ra	ri	ru	re	ro	ら	り	る	れ	ろ	ラ	リ	ル	レ	ロ
la	li	lu	le	lo	ら	り	る	れ	ろ	ラ	リ	ル	レ	ロ
wa	wi	wu	we	wo	わ	ゐ	う	ゑ	を	ワ	ヰ	ウ	ヱ	ヲ
nn					ん					ン				
ga	gi	gu	ge	go	が	ぎ	ぐ	げ	ご	ガ	ギ	グ	ゲ	ゴ
za	zi	zu	ze	zo	ざ	じ	ず	ぜ	ぞ	ザ	ジ	ズ	ゼ	ゾ
	ji					じ					ジ			
da	di	du	de	do	だ	ぢ	づ	で	ど	ダ	ヂ	ヅ	デ	ド
ba	bi	bu	be	bo	ば	び	ぶ	べ	ぼ	バ	ビ	ブ	ベ	ボ
pa	pi	pu	pe	po	ぱ	ぴ	ぷ	ぺ	ぽ	パ	ピ	プ	ペ	ポ
kya	kyi	kyu	kye	kyo	きゃ	きぃ	きゅ	きぇ	きょ	キャ	キィ	キュ	キェ	キョ
sha		shu	she	sho	しゃ		しゅ	しぇ	しょ	シャ		シュ	シェ	ショ
sya	syi	syu	sye	syo	しゃ	しぃ	しゅ	しぇ	しょ	シャ	シィ	シュ	シェ	ショ
cha		chu	che	cho	ちゃ		ちゅ	ちぇ	ちょ	チャ		チュ	チェ	チョ
tya	tyi	tyu	tye	tyo	ちゃ	ちぃ	ちゅ	ちぇ	ちょ	チャ	チィ	チュ	チェ	チョ
cya	cyi	cyu	cye	cyo	ちゃ	ちぃ	ちゅ	ちぇ	ちょ	チャ	チィ	チュ	チェ	チョ
nya	nyi	nyu	nye	nyo	にゃ	にぃ	にゅ	にぇ	にょ	ニャ	ニィ	ニュ	ニェ	ニョ
hya	hyi	nyu	nye	nyo	ひぁ	ひぃ	ひゅ	ひぇ	ひょ	ヒャ	ヒィ	ヒュ	ヒェ	ヒョ
mya	myi	myu	mye	myo	みゃ	みぃ	みゅ	みぇ	みょ	ミャ	ミィ	ミュ	ミェ	ミョ
rya	ryi	ryu	rye	ryo	りゃ	りぃ	りゅ	りぇ	りょ	リャ	リィ	リュ	リェ	リョ
lya	lyi	lyu	lye	lyo	りゃ	りぃ	りゅ	りぇ	りょ	リャ	リィ	リュ	リェ	リョ
gya	gyi	gyu	gye	gyo	ぎゃ	ぎぃ	ぎゅ	ぎぇ	ぎょ	ギャ	ギィ	ギュ	ギェ	ギョ
ja		ju	je	jo	じゃ		じゅ	じぇ	じょ	ジャ		ジュ	ジェ	ジョ
jya	jyi	jyu	jye	jyo	じゃ	じぃ	じゅ	じぇ	じょ	ジャ	ジィ	ジュ	ジェ	ジョ
zya	zyi	zyu	zye	zyo	じゃ	じぃ	じゅ	じぇ	じょ	ジャ	ジィ	ジュ	ジェ	ジョ
dya	dyi	dyu	dye	dyo	ぢゃ	ぢぃ	ぢゅ	ぢぇ	ぢょ	ヂャ	ヂィ	ヂュ	ヂェ	ヂョ
bya	byi	byu	bye	byo	びゃ	びぃ	びゅ	びぇ	びょ	ビャ	ビィ	ビュ	ビェ	ビョ
pya	pyi	pyu	pye	pyo	ぴゃ	ぴぃ	ぴゅ	ぴぇ	ぴょ	ピャ	ピィ	ピュ	ピェ	ピョ
gwa	gwi	gwu	gwe	gwo	ぐゎ	ぐぃ	ぐぅ	ぐぇ	ぐぉ	グヮ	グィ	グゥ	グェ	グォ
qwa	qwi	qwu	qwe	qwo	くゎ	くぃ	くぅ	くぇ	くぉ	クヮ	クィ	クゥ	クェ	クォ
kwa	kwi	kwu	kwe	kwo	くゎ	くぃ	くぅ	くぇ	くぉ	クヮ	クィ	クゥ	クェ	クォ
fa	fi		fe	fo	ふぁ	ふぃ		ふぇ	ふぉ	ファ	フィ		フェ	フォ
va	vi	vu	ve	vo	ゔぁ	ゔぃ	ぶ	ゔぇ	ゔぉ	ヴァ	ヴィ	ヴ	ヴェ	ヴォ
qa	qi		qe	qo	くぁ	くぃ		くぇ	くぉ	クァ	クィ		クェ	クォ

ローマーかな対応表（２）

ローマ字	ひらがな	カタカナ
tsa tsi tse tso	つぁ つぃ つぇ つぉ	ツァ ツィ ツェ ツォ
xa xi xu xe xo	ぁ ぃ う ぇ ぉ	ア イ ウ エ オ
xya xyu xyo	や ゆ よ	ヤ ユ ヨ
xka xke xca		カ ケ カ
xtu xtsu	っ っ	ッ ッ
xwa	わ	ワ

・子音の読み（後に母音または同じ子音が続く場合を除く、カタカナも同じ）

b ：ぶ by：び	c ：く cy；ち ch；ち	d ：ど dy：ぢ	f ：ふ	g ：ぐ gy：ぎ gw：ぐゎ	h ：は hy：ひ	j ：じ jy：じ	k ：く ky：き kw：くゎ
l ：る ly：り	m ：む my；み	n ：ど ny：ぢ	p ：ぷ py：ぴ	q ：く qw：くゎ	r ：る ry：り	s ：す sh：しゅ sy：し	t ：と tch：とち ts ：つ ty ：ち
v ：ぶ ：(ヴ)	w ：わ	y ：い	z ：ず zy：じ				

F. 記号読み入力の一覧表

あくせんと	´ ¨
いっぱんきごう	§ ※ 〒 ㈱ № ℡ ‥ …
えんざんきごう	― ± ≠ ∞ - ≦ ∴ ≧ ∵ × ÷
かっこきごう	‘ “ 〔 〈 《 【 ’ ” 〕 〉 》 】
ぎりしゃもじ	α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω
くろきごう	▲ ● ▲ ★ ◆ ■ ▼
こんざいきごう	├ ┬ ┤ ┴ ┼ ├ ┬ ┤ ┴ ┼
しろきごう	○ △ ◎ ☆ ◇ □ ▽
すうがくきごう	∠ ⊥ ⌒ ∂ ∇ ≡ ≒ ≪ ≫ √ ∽ ∝ ∫ ∬
たんいきごう	¢ ℃ ° ′ ″ Å ‰
とくしゅきごう	¦ 〃 仝 ♂ ♀ ‖ ゠ ♯ ♭ ♪ † ‡ ¶ ○
ふとせんきごう	━ ┃ ┏ ┓ ┛ ┗ ┣ ┳ ┫ ┻ ╋
ふるいかたかな	ヷ ヵ ㇰ ㇱ ㇲ ㇳ ㇴ
ふるいひらがな	ゎ ゐ ゑ ゝ ゞ
ほそせんきごう	―	┌ ┐ ┘ └ ├ ┬ ┤ ┴ ┼
やじるし	→ ← ↑ ↓ ⇒ ⇔
ろおますうじ	i ii iii iv v vi vii viii ix x Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ
ろしあもじ	а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я
ろんりきごう	∈ ∋ ⊆ ⊇ ⊂ ⊃ ∪ ∩ ∧ ∨ ∀ ∃

G. ターミナル・エミュレーション

1. Windows ターミナル・エミュレーション

AIX V4.3.2 から TERMINFO vt100-msjp により、Windows 95, 98, NT の PC が AIX の端末としてサポートされていましたが、AIX V5.3 より提供される新しい TERMINFO ansi-msjp により、Windows 2000, XP の PC がサポートされます。

以下のような Windows のターミナル・エミュレーションが使用可能となります。

- Windows の telnet プログラムを使って AIX へログインし、そこで漢字を入力、表示。
- シリアル・ポートから tty 接続して AIX へログインし、そこで漢字を入力、表示。

使用手順は次の通りです。

① Windows マシンから AIX のホストマシンへ telnet/tty でログインします。
② ログイン画面から、以下の環境変数を設定してください。

```
export TERM=ansi-msjp
```

2. HMC ターミナル・エミュレーション

AIX で提供される TERMINFO iterm により、ハードウェア管理コンソール(HMC, IBM Hardware Management Console)の日本語入力機能ががサポートされます。AIX の以下のリリースおよび、前提 APAR の適用により、HMC 上の端末エミュレーターで日本語入力が可能になります。

AIX: V5.1 APAR=IY54785, V5.2 APAR=IY54562, V5.3以降
HMC: HMCライセンス機械コード V4 R3.0以降

使用手順は次の通りです。

① ハードウェア管理コンソールのコンソール・メニューから「端末エミュレーターを開く」を選んで、ホストマシンへログインします。
② ログイン画面から、以下の環境変数を設定してください。

```
export TERM=iterm
export LANG=ja_JP
```

注）Windows や HMC のターミナル・エミュレーションは telnet セッションを使用するため、AIX 日本語入力機能は使用されず、Windows や HMC の日本語入力機能が使用されます。

H. プログラム・リスト immsample.c

```
/*
 * sample program - immsample.c
 */
/*****************/
/* include files */
/*****************/
#include <stdio.h>
#include <locale.h>
#include <Xm/Xm.h>
#include <Xm/RowColumn.h>
#include <Xm/PushB.h>
#include <Xm/Text.h>

/***********************/
/* global declarations */
/***********************/
void bye(Widget w, caddr_t client_data, caddr_t call_data);

/****************/
/* main program */
/****************/
int main(int argc, char **argv)
{
    /*******************/
    /* local variables */
    /*******************/
    XtAppContext  app;
    Widget        toplevel;
    Widget        rc;
    Widget        exitbutton;
    Widget        textwidget;
    Arg           args[10];
    int           n;

    /*******************************/
    /* set locale (言語環境の確立) */
    /*******************************/
    XtSetLanguageProc(NULL, (XtLanguageProc)NULL, NULL);

    /***************************************************/
    /* initialize toolkit (イントリンシックスの初期化) */
    /***************************************************/
    toplevel= XtAppInitialize(&app, "Immsample", (XrmOptionDescList)NULL, 0,
                              &argc, argv, (String *)NULL, (ArgList)NULL, 0);

    /*****************************************/
    /* create widgets (各ウィジェットの生成) */
    /*****************************************/
    n = 0;
    rc = XtCreateManagedWidget("Rc",
                        xmRowColumnWidgetClass, toplevel, args, n);

    n = 0;
    exitbutton = XtCreateManagedWidget("Button",
                        xmPushButtonWidgetClass, rc, args, n);

    /*****************************************/
    /* add callback (コールバック手続きの定義) */
    /*****************************************/
    XtAddCallback(exitbutton, XmNactivateCallback, bye, (caddr_t)NULL);
```

```
    n = 0;
    XtSetArg(args[n], XmNeditMode, XmMULTI_LINE_EDIT); n++;
    XtSetArg(args[n], XmNrows, 5); n++;
    XtSetArg(args[n], XmNcolumns, 15); n++;
    textwidget = XtCreateManagedWidget("Text",
                        xmTextWidgetClass, rc, args, n);

    /*************************************/
    /* realize widget (ウィジェットの表示)  */
    /*************************************/
    XtRealizeWidget(toplevel);
    /*************************************/
    /* main loop (イベント・ループへ入る)   */
    /*************************************/
    XtAppMainLoop(app);
}

/*************************************/
/* callback routine for button widget */
/*************************************/
void bye(Widget w, caddr_t client_data, caddr_t call_data)
{
    fprintf(stderr, "bye callback is called.¥n");
    exit(0);
}
```

I. プログラム・リスト ximsample.c

```
/************************************************************************/
/*                                                                      */
/* サンプルプログラム：ximsample.c                                      */
/*                                                                      */
/* このサンプルプログラムは、X11R6 でサポートされている、XIM API を     */
/* 用いて、テキスト入力を行う一例を示します。                           */
/*                                                                      */
/* 日本語環境で実行するには、次のコマンドを実行して下さい。             */
/*      ximsample -lang ja_JP                                           */
/*                                                                      */
/* 英語環境で実行するには、次のコマンドを実行して下さい。               */
/*      ximsample -lang en_US -nothing                                  */
/*                                                                      */
/* 実行すると、クライアント・ウィンドウが現れ、左上に四角いカーソルが   */
/* 表示されます。適当なテキスト入力を行って下さい。                     */
/*                                                                      */
/* 終了する時は、クライアント・ウィンドウ上で任意のマウス・ボタンを     */
/* 押して下さい。                                                       */
/*                                                                      */
/************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>          /* XSizeHints 構造体を使用する。        */
#include <X11/keysym.h>         /* キーシムを使用する。                 */

#define CLIENT_MARGIN   5       /* クライアントウィンドウ上のマージン   */
#define CLIENT_BORDER   5       /* クライアントウィンドウの枠幅         */
#define FOCUS_BORDER    1       /* フォーカスウィンドウの枠幅           */
#define MIN(x,y)        (((x) < (y)) ? (x) : (y))


int
main( int argc, char *argv[] )
{
        Display         *display;
        int             screen;
        XIMStyle        input_style = XIMPreeditPosition|XIMStatusArea;
                                        /* 希望する入力スタイル         */
        XIMStyles       *im_styles;     /* XIM がサポートしているスタイル*/
        XFontSet        fontset;        /* 文字表示用フォントセット     */
        XRectangle      ink, log;       /* 文字列の矩形領域情報         */
        XSizeHints      normal_hints;   /* ウィンドウサイズのヒント     */
        Window          client_window, focus_window;    /* ウィンドウ   */
        XIM             xim;            /* Xインプット・メソッド        */
        XIC             xic;            /* Xインプット・コンテクスト    */
        GC              gc;             /* グラフィック・コンテクスト   */
        Colormap        cmap;           /* カラーマップ                 */
        XColor          color;          /* カラー資源識別子             */
        unsigned long   fg, bg, black, white;   /* ピクセル値           */
        char            *base_name = "-dt-interface user-medium-r-normal-l*";
                                        /* ベースフォント名             */
        char            *disp_name = NULL;      /* ディスプレー名       */
        char            *fg_name = "black";     /* 前景色名             */
        char            *bg_name = "white";     /* 背景色名             */
        char            *locale = NULL;         /* ロケール名           */
        char            *modifiers = NULL;      /* モディファイア名     */
        char            **missing_list;         /* ミッシング・リスト   */
```

```
char            *def_string;            /* デフォルト文字列     */
char            *invalid_arg;           /* 無効な引数名         */
char            window_name[64], icon_name[64];
int             missing_count;          /* ミッシング・カウント */
int             num_col=60;             /* カラム数             */
int             num_row=20;             /* 行数                 */
int             client_width, client_height;
int             focus_width, focus_height;
int             i;
Bool            found;


/*
 * コマンドライン・オプションの解析をし、各変数に設定する。
 */
for (i=1; i<argc; i++) {
        if (!strcmp(argv[i],"-root")) {
                input_style = XIMPreeditNothing|XIMStatusNothing;
        } else if (!strcmp(argv[i],"-over")) {
                input_style = XIMPreeditPosition|XIMStatusArea;
        } else if (!strcmp(argv[i],"-off")) {
                input_style = XIMPreeditArea|XIMStatusArea;
        } else if (!strcmp(argv[i],"-nothing")) {
                input_style = XIMPreeditNothing|XIMStatusNone;
        } else if (!strcmp(argv[i],"-none")) {
                input_style = XIMPreeditNone|XIMStatusNone;
        } else if (!strcmp(argv[i],"-fs")) {
                base_name = argv[++i];
        } else if (!strcmp(argv[i],"-lang")) {
                locale = argv[++i];
        } else if (!strcmp(argv[i],"-mod")) {
                modifiers = argv[++i];
        } else if (!strcmp(argv[i],"-display")) {
                disp_name = argv[++i];
        } else if (!strcmp(argv[i],"-fg")) {
                fg_name = argv[++i];
        } else if (!strcmp(argv[i],"-bg")) {
                bg_name = argv[++i];
        } else {
                usage(argv);
                exit(1);
        }
}

/*
 * システムの言語環境（ロケール）の設定
 */
if (locale == (char *)NULL) {
        locale = setlocale(LC_ALL,"");
} else {
        locale = setlocale(LC_ALL,locale);
}
if (locale == (char *)NULL || *locale == (char)NULL) {
        fprintf(stderr,"Can't set locale¥n");
        return (1);
}
```

```
/*
 * 上で設定を行った言語環境がX-Windowでサポートされているかを
 * 調べる。
 */
if (!XSupportsLocale()) {
        fprintf(stderr,"locale [%s] is not supported by X¥n",
                setlocale(LC_CTYPE,NULL));
        return (1);
}

/*
 * ロケール・モディファイアを設定する。
 */
modifiers = XSetLocaleModifiers(modifiers);

/*
 * Xサーバーとの接続を行う。
 */
display = XOpenDisplay(disp_name);
if (display == (Display *)NULL) {
        fprintf(stderr,"Can't open display %s¥n",disp_name);
        return (1);
}
screen = DefaultScreen(display);

/*
 * フォントセットを作成する。
 */
fontset = XCreateFontSet(display,base_name,
                         &missing_list,
                         &missing_count,
                         &def_string);

/*
 * 見つからなかったキャラクターセットを表示させる。
 */
if (missing_count > 0) {
        fprintf(stderr,"Missing charsets are detected¥n");
        for (i=0; i<missing_count; i++) {
                fprintf(stderr,"¥t%d : [%s]¥n",i,missing_list[i]);
        }
        XFreeStringList(missing_list);
}
if (fontset == (XFontSet)NULL) {
        fprintf(stderr,"Can't create fontset [%s]¥n",base_name);
        return (1);
}

/*
 * フォントセットの大きさ情報を用いて、クライアント・ウィンドウ
 * フォーカス・ウィンドウの大きさを求める。
 */
XmbTextExtents(fontset,"#",1,&ink,&log);
focus_width = num_col * log.width;
focus_height = num_row * log.height;
client_width = focus_width + 2 * CLIENT_MARGIN;
client_height = focus_height + 2 * CLIENT_MARGIN;
```

```
/*
 * 指定された色を用意する。
 */
black = BlackPixel(display,screen);
white = WhitePixel(display,screen);
cmap = DefaultColormap(display,screen);
if(XParseColor(display,cmap,fg_name,&color) == 0) {
        fprintf(stderr,"Bad color specification [%s]¥n",fg_name);
        fg = black;
} else {
        if (XAllocColor(display,cmap,&color) == 0) {
                fprintf(stderr,"Can't allocale color¥n");
                fg = black;
        } else {
                fg = color.pixel;
        }
}
if(XParseColor(display,cmap,bg_name,&color) == 0) {
        fprintf(stderr,"Bad color specification [%s]¥n",bg_name);
        bg = white;
} else {
        if (XAllocColor(display,cmap,&color) == 0) {
                fprintf(stderr,"Can't allocale color¥n");
                bg = white;
        } else {
                bg = color.pixel;
        }
}

/*
 * クライアント・ウィンドウとフォーカス・ウィンドウを作成する。
 */
client_window = XCreateSimpleWindow(display,
                                    RootWindow(display,screen),
                                    0,0,client_width,client_height,
                                    CLIENT_BORDER,black,white);
XSelectInput(display,client_window,
             StructureNotifyMask|FocusChangeMask|ButtonPressMask);
focus_window = XCreateSimpleWindow(display,
                                   client_window,
                                   CLIENT_MARGIN-FOCUS_BORDER,
                                   CLIENT_MARGIN,
                                   focus_width,focus_height,
                                   FOCUS_BORDER,black,white);

/*
 * ＧＣを作成する。
 */
gc = XCreateGC(display,focus_window,0L,NULL);


/*
 * Ｘインプット・メソッドと接続する。（リソースの指定無しで）
 */
xim = XOpenIM(display,NULL,NULL,NULL);
if (xim == (XIM)NULL) {
        fprintf(stderr,"Can't open XIM¥n");
        return (1);
}
```

```
/*
 * Xインプット・メソッドがサポートしている入力スタイルを調べる。
 */
invalid_arg = XGetIMValues(xim,XNQueryInputStyle,&im_styles,NULL);
if (invalid_arg != (char *)NULL) {
        fprintf(stderr,"Invalid argument [%s]¥n",invalid_arg);
        return (1);
}

/*
 * 希望する入力スタイルがXインプット・メソッドでサポートされて
 * いるかどうかを調べる。
 */
found = False;
if (im_styles != (XIMStyles *)NULL) {
        for (i=0; i<im_styles->count_styles; i++) {
                if (input_style == im_styles->supported_styles[i]) {
                        /* 見つけた */
                        found = True;
                        break;
                }
        }
} else {
        fprintf(stderr,"Can't get input styles¥n");
        return (1);
}
if (!found) {
        fprintf(stderr,"Unsupported input style¥n");
        return (1);
}


/*
 * Xインプット・コンテクストを作成する。
 * 入力スタイルによって、作成の引き数が異なるので注意すること。
 */
if (input_style == (XIMPreeditPosition|XIMStatusArea)) {
        XPoint          spot;
        XRectangle      area;
        XFontSetExtents *fs_ext = XExtentsOfFontSet(fontset);
        XVaNestedList   preedit_attr, status_attr;

        /* XNSpotLocationの指定が必要 */
        spot.x = fs_ext->max_logical_extent.x;
        spot.y = -(fs_ext->max_logical_extent.y);
        area.x = 0;
        area.y = 0;
        area.width = focus_width;
        area.height = focus_height;
        preedit_attr = XVaCreateNestedList(0,
                                           XNFontSet,fontset,
                                           XNForeground,fg,
                                           XNBackground,bg,
                                           XNSpotLocation,&spot,
                                           XNArea,&area,
                                           NULL);
        status_attr = XVaCreateNestedList(0,
                                          XNFontSet,fontset,
                                          XNForeground,fg,
                                          XNBackground,bg,
                                          NULL);
```

```
        xic = XCreateIC(xim,
                        XNInputStyle,input_style,
                        XNClientWindow,client_window,
                        XNFocusWindow,focus_window,
                        XNPreeditAttributes,preedit_attr,
                        XNStatusAttributes,status_attr,
                        NULL);
        XFree(preedit_attr);
        XFree(status_attr);

    } else if (input_style == (XIMPreeditArea|XIMStatusArea)) {
        XVaNestedList   preedit_attr, status_attr;

        preedit_attr = XVaCreateNestedList(0,
                                           XNFontSet,fontset,
                                           XNForeground,fg,
                                           XNBackground,bg,
                                           NULL);
        status_attr = XVaCreateNestedList(0,
                                          XNFontSet,fontset,
                                          XNForeground,fg,
                                          XNBackground,bg,
                                          NULL);
        xic = XCreateIC(xim,
                        XNInputStyle,input_style,
                        XNClientWindow,client_window,
                        XNFocusWindow,focus_window,
                        XNPreeditAttributes,preedit_attr,
                        XNStatusAttributes,status_attr,
                        NULL);
        XFree(preedit_attr);
        XFree(status_attr);

    } else if (input_style == (XIMPreeditNothing|XIMStatusNothing)) {
        XVaNestedList   preedit_attr, status_attr;

        preedit_attr = XVaCreateNestedList(0,
                                           XNFontSet,fontset,
                                           XNForeground,fg,
                                           XNBackground,bg,
                                           NULL);
        status_attr = XVaCreateNestedList(0,
                                          XNFontSet,fontset,
                                          XNForeground,fg,
                                          XNBackground,bg,
                                          NULL);
        /* XNClientWindow の指定は必要ない */
        xic = XCreateIC(xim,
                        XNInputStyle,input_style,
                        XNClientWindow,client_window,
                        XNFocusWindow,focus_window,
                        XNPreeditAttributes,preedit_attr,
                        XNStatusAttributes,status_attr,
                        NULL);
        XFree(preedit_attr);
        XFree(status_attr);

    } else if (input_style == (XIMPreeditNothing|XIMStatusNone)) {
        XPoint          spot;
        XVaNestedList   preedit_attr;
```

```
        preedit_attr = XVaCreateNestedList(0,
                                           XNFontSet,fontset,
                                           XNForeground,fg,
                                           XNBackground,bg,
                                           XNSpotLocation,&spot,
                                           NULL);
        xic = XCreateIC(xim,
                        XNInputStyle,input_style,
                        XNClientWindow,client_window,
                        XNFocusWindow,focus_window,
                        XNPreeditAttributes,preedit_attr,
                        NULL);
        XFree(preedit_attr);

} else if (input_style == (XIMPreeditNone|XIMStatusNone)) {
        XPoint          spot;
        XVaNestedList   preedit_attr;

        preedit_attr = XVaCreateNestedList(0,
                                           XNFontSet,fontset,
                                           XNForeground,fg,
                                           XNBackground,bg,
                                           XNSpotLocation,&spot,
                                           NULL);
        xic = XCreateIC(xim,
                        XNInputStyle,input_style,
                        XNClientWindow,client_window,
                        XNFocusWindow,focus_window,
                        XNPreeditAttributes,preedit_attr,
                        NULL);
        XFree(preedit_attr);
}
if (xic == (XIC)NULL) {
        fprintf(stderr,"Can't create input context¥n");
        return (1);
}


/*
 * XIMStatusArea の場合、適切なステータス領域の大きさを求め、
 * 設定する。
 */
if (input_style & XIMStatusArea) {
        XVaNestedList   status_attr;
        XRectangle      *area_needed, area;
        unsigned long   filter_event;

        /* インプット・メソッドが必要とするステータス領域の     */
        /* 大きさを求める。                                     */
        /* 同時に、インプット・メソッドが必要とするイベントの   */
        /* マスク値も求める。                                   */
        status_attr = XVaCreateNestedList(0,
                                          XNAreaNeeded,&area_needed,
                                          NULL);
        invalid_arg = XGetICValues(xic,
                                   XNFilterEvents,&filter_event,
                                   XNStatusAttributes,status_attr,
                                   NULL);
        if (invalid_arg != (char *)NULL) {
                fprintf(stderr,"Invalid argument [%s]¥n",invalid_arg);
                return (1);
        }
        XFree(status_attr);
```

```
        /* フォーカス・ウィンドウの大きさを考慮して適切な       */
        /* ステータス領域の大きさを決定する。                   */
        area.x = CLIENT_MARGIN;
        area.y = focus_height + 2 * CLIENT_MARGIN;
        area.width = MIN(area_needed->width,focus_width);
        area.height = area_needed->height;

        /* インプット・メソッドに決定したステータス領域の大きさ */
        /* を設定する。                                         */
        status_attr = XVaCreateNestedList(0,
                                        XNArea,&area,
                                        NULL);
        invalid_arg = XSetICValues(xic,
                                XNStatusAttributes,status_attr,
                                NULL);
        if (invalid_arg != (char *)NULL) {
                fprintf(stderr,"Invalid argument [%s]¥n",invalid_arg);
                return (1);
        }
        XFree(status_attr);

        /* クライアント・プログラムが必要とするイベント・マスク */
        /* にインプット・メソッドが必要とするイベント・マスクを */
        /* 加えたものをフォーカス・ウィンドウに設定する。       */
        XSelectInput(display,focus_window,ExposureMask|KeyPressMask|
                     FocusChangeMask|filter_event);

        /* クライアント・ウィンドウの大きさを変える。           */
        client_height += area.height + CLIENT_MARGIN;
        XResizeWindow(display,client_window,
                      client_width,client_height);
} else {
        unsigned long   filter_event;

        /* インプット・メソッドが必要とするイベント・マスク     */
        /* を求める。                                           */
        invalid_arg = XGetICValues(xic,
                                XNFilterEvents,&filter_event,
                                NULL);
        if (invalid_arg != (char *)NULL) {
                fprintf(stderr,"Invalid argument [%s]¥n",invalid_arg);
                return (1);
        }

        /* クライアント・プログラムが必要とするイベント・マスク */
        /* にインプット・メソッドが必要とするイベント・マスクを */
        /* 加えたものをフォーカス・ウィンドウに設定する。       */
        XSelectInput(display,focus_window,ExposureMask|KeyPressMask|
                     FocusChangeMask|filter_event);
}
```

```
/*
 * XIMPreeditArea の場合(OffTheSpot)、プリエディット領域をステータス領域の右側にする。
 *
 *      +----------------------------------+
 *      |+--------------------------------+|
 *      ||                                ||<== Client Window
 *      ||                                ||
 *      ||                                ||
 *      ||          Focus Window          ||
 *      ||                                ||
 *      ||                                ||
 *      ||                                ||
 *      |+--------------------------------+|
 *      |+--------++----------------------+|
 *      || Status ||     Preedit Area     ||
 *      |+--------++----------------------+|
 *      +----------------------------------+
 *
 */
if (input_style & XIMPreeditArea) {
        XVaNestedList   preedit_attr, status_attr;
        XRectangle      preedit_area, *status_area;

        /* 既に設定されているステータス領域の大きさを調べる。    */
        status_attr = XVaCreateNestedList(0,
                                        XNArea,&status_area,
                                        NULL);
        invalid_arg = XGetICValues(xic,
                                XNStatusAttributes,status_attr,
                                NULL);
        if (invalid_arg != (char *)NULL) {
                fprintf(stderr,"Invalid argument [%s]¥n",invalid_arg);
                return (1);
        }
        XFree(status_attr);

        /* フォーカス・ウィンドウの大きさと上で求めたステータス */
        /* 領域の大きさから、プリエディット領域の大きさを計算   */
        /* する。                                              */
        preedit_area.x = status_area->x + status_area->width;
        preedit_area.y = status_area->y;
        preedit_area.width = focus_width - status_area->width;
        preedit_area.height = status_area->height;

        /* 計算したプリエディット領域の大きさを設定する。        */
        preedit_attr = XVaCreateNestedList(0,
                                        XNArea,&preedit_area,
                                        NULL);
        invalid_arg = XSetICValues(xic,
                                XNPreeditAttributes,preedit_attr,
                                NULL);
        if (invalid_arg != (char *)NULL) {
                fprintf(stderr,"Invalid argument [%s]¥n",invalid_arg);
                return (1);
        }
        XFree(preedit_attr);
}

/*
 * ウィンドウ・マネージャ用の各種プロパティを設定する。
 * ウィンドウ名、アイコン名、ウィンドウの大きさ等。
 */
sprintf(window_name,"%s [%s]¥n",argv[0],setlocale(LC_CTYPE,NULL));
```

```
    sprintf(icon_name,"%s icon¥n",argv[0]);
    normal_hints.flags = PMinSize | PMaxSize;
    normal_hints.min_width = client_width;
    normal_hints.max_width = client_width;
    normal_hints.min_height = client_height;
    normal_hints.max_height = client_height;
    XmbSetWMProperties(display,client_window,window_name,icon_name,
                       argv,argc,&normal_hints,
                       (XWMHints *)NULL,(XClassHint *)NULL);

    /*
     * テキスト・バッファの初期化
     */
    InitText(display,fontset,num_row,num_col);
    XMapWindow(display,client_window);
    XMapWindow(display,focus_window);


    /*
     * メイン・イベント・ループ
     */
    while(1) {
            XEvent  event;
            char    buffer[1024];
            int     buf_len = 1024;
            int     len;
            KeySym  keysym = NoSymbol;
            Status  status;
            XPoint  spot;

            /* イベントを取り込み、インプット・メソッドに渡す。      */
            /* もし、そのイベントがインプット・メソッドに使用された */
            /* 場合は、XFilterEvent()はTrueを返すので、イベント     */
            /* ループを繰り返す。もし、そのイベントがインプット・    */
            /* メソッドに使用されなかった場合は、クライアント側の    */
            /* 処理を続ける。                                      */
            XNextEvent(display,&event);
            if (XFilterEvent(&event,(Window)NULL)) {
                    continue;
            }

            /* イベント・タイプに応じた処理を行わせる。              */
            switch (event.type) {
            case ButtonPress :
                    /* マウス・ボタンが押されたら、終了。            */
                    Quit(display,xim,xic,fontset,gc,
                         client_window,focus_window);
                    break;
            case FocusIn :
                    /* キーボード・フォーカスが当たった。            */
                    XSetInputFocus(display,focus_window,
                                   RevertToParent,CurrentTime);
                    XSetICFocus(xic);
                    break;
            case FocusOut :
                    /* キーボード・フォーカスが失われた。            */
                    XUnsetICFocus(xic);
                    break;
            case Expose :
                    /* Exposeイベント、テキスト・バッファの内容を    */
                    /* 再描画する。                                 */
                    if (event.xany.window == focus_window) {
                            RedisplayText(display,focus_window,
```

```
                                fontset,gc,&event);
        }
        break;
case KeyPress :
        /* キーが押された。確定文字列を取得する。        */
        len = XmbLookupString(xic,(XKeyPressedEvent *)&event,
                                buffer,buf_len,&keysym,&status);
        if (status == XBufferOverflow) {
                /* 指定したバッファが足りない。          */
                fprintf(stderr,
                        "Buffer overflow %d bytes needed¥n",
                        len);
        } else if (status == XLookupNone) {
                /* 確定文字列もキーシムも取得できない。 */
                /* Nothing returned */
                /* Nothing to do */
        } else if (status == XLookupChars) {
                /* 確定文字列だけ取得できた。            */
                XVaNestedList   preedit_attr;

                /* 確定文字列の長さが0 */
                if (len <= 0) {
                        break;
                }
                /* 確定文字列を描画する。                */
                WriteText(display,focus_window,fontset,gc,
                          buffer,len,NoSymbol,
                          &(spot.x),&(spot.y));

                /* スポット・ロケーションを更新する。    */
                preedit_attr =
                        XVaCreateNestedList(0,
                                            XNSpotLocation,
                                            &spot,
                                            NULL);
                XSetICValues(xic,
                             XNPreeditAttributes,preedit_attr,
                             NULL);
                XFree(preedit_attr);
        } else if (status == XLookupKeySym) {
                /* キーシムだけ取得できた。              */
                XVaNestedList   preedit_attr;
                XPoint          old_spot;

                /* キーシムの種類を調べる。            */
                /* 表示不可能なキーシムならブレーク    */
                if (IsCursorKey(keysym) ||
                    IsFunctionKey(keysym) ||
                    IsKeypadKey(keysym) ||
                    IsMiscFunctionKey(keysym) ||
                    IsModifierKey(keysym) ||
                    IsPFKey(keysym)) {
                        break;
                }

                /* 改行などを行う。                      */
                old_spot.x = spot.x;
                old_spot.y = spot.y;
                WriteText(display,focus_window,fontset,gc,
                          NULL,0,keysym,
                          &(spot.x),&(spot.y));

                /* スポット・ロケーションを更新する。    */
```

```
                        if (old_spot.x != spot.x ||
                            old_spot.y != spot.y ) {
                                preedit_attr =
                                        XVaCreateNestedList(0,
                                                            XNSpotLocation,
                                                            &spot,
                                                            NULL);
                                XSetICValues(xic,
                                     XNPreeditAttributes,preedit_attr,
                                     NULL);
                                XFree(preedit_attr);
                        }
                } else if (status == XLookupBoth) {
                        /* 確定文字列とキーシムが取得できた。    */
                        XVaNestedList   preedit_attr;

                        /* 確定文字列の長さが０ */
                        if (len <= 0) {
                                break;
                        }
                        /* 確定文字列を描画し、改行などを行う。 */
                        WriteText(display,focus_window,fontset,gc,
                                  buffer,len,keysym,
                                  &(spot.x),&(spot.y));

                        /* スポット・ロケーションを更新する。    */
                        preedit_attr =
                                XVaCreateNestedList(0,
                                                    XNSpotLocation,
                                                    &spot,
                                                    NULL);
                        XSetICValues(xic,
                                     XNPreeditAttributes,preedit_attr,
                                     NULL);
                        XFree(preedit_attr);
                } else {
                        fprintf(stderr,"Unknown status is returned¥n");
                }
                break;
        case DestroyNotify :
                /* ウィンドウが破壊された。プログラムを終了。    */
                Quit(display,xim,xic,fontset,gc,
                     client_window,focus_window);
                break;
        case MappingNotify :
                /* キーボード・マッピングが変更された。          */
                XRefreshKeyboardMapping((XMappingEvent *)&event);
                break;
        }
    }
}
```

```
/*
 *      Quit()  : プログラムを終了させる。
 */
static  int
Quit( Display *display, XIM xim, XIC xic, XFontSet fontset, GC gc,
      Window client_window, Window focus_window )
{
        if (xic) XDestroyIC(xic);
        if (xim) XCloseIM(xim);
        if (gc) XFreeGC(display,gc);
        if (display && focus_window) XDestroyWindow(display,focus_window);
        if (display && client_window) XDestroyWindow(display,client_window);
        if (display && fontset) XFreeFontSet(display,fontset);
        if (display) XCloseDisplay(display);
        exit (0);
}

/*
 *      テキスト・バッファを扱うためのLineBuffer 構造体
 */
typedef struct  _LineBufferRec  {
        char    *string;
        int     len;
        int     max_size;
} LineBufferRec, *LineBuffer;

/*
 *      テキスト・バッファ操作関数で用いる、グローバル変数。
 */
static  LineBuffer      TextBuffer = (LineBuffer)NULL;
static  int             TextLines = 0;
static  int             CurrentLine = 0;
static  int             FontWidth, FontHeight, FontY;
static  int             TextRow, TextCol;


/*
 *      InitText()      : テキスト・バッファの初期化
 */
static  int
InitText( Display *display, XFontSet fontset, int row, int col)
{
        XRectangle      ink, log;
        int             i;

        /* LineBuffer を作る。                                          */
        TextBuffer = (LineBuffer)calloc(row,sizeof(LineBufferRec));
        if (TextBuffer == (LineBuffer)NULL) {
                fprintf(stderr,"Can't allocate memory¥n");
                exit (1);
        }
        TextLines = row;

        /* LineBuffer の各ラインを初期化する。                          */
        for (i=0; i<TextLines; i++) {
                char    *string;
                string = (char *)calloc(1,MB_CUR_MAX*col);
                if (string == (char *)NULL) {
                        fprintf(stderr,"Can't allocate memory¥n");
                        exit (1);
                }
                TextBuffer[i].string = string;
                TextBuffer[i].max_size = MB_CUR_MAX*col;
```

```
        }

        /* グローバル変数の初期化。                                        */
        XmbTextExtents(fontset,"#",1,&ink,&log);
        FontWidth = log.width;
        FontHeight = log.height;
        FontY = -(log.y);
        TextRow = row;
        TextCol = col;
        return (0);
}

/*
 *      WriteText()     : テキスト・バッファの更新と描画
 */
static  int
WriteText( Display *display, Window window, XFontSet fontset, GC gc,
           char *string, int len, KeySym keysym, short *x, short *y )
{
        static Bool     initialized = False;
        XGCValues       gc_values;

        /* 初期化 */
        if (!initialized) {
                *x = 0;
                *y = FontY;
                initialized = True;
        }

        /* 扱うキーシムはリターンとタブのみに限定                          */
        if ((keysym != XK_Return || keysym != XK_Tab) &&
            len < 1) {
                return (0);
        }

        /* 古いカーソルを消去する。                                        */
        XGetGCValues(display,gc,GCForeground|GCBackground,&gc_values);
        XSetForeground(display,gc,gc_values.background);
        XSetBackground(display,gc,gc_values.foreground);
        XFillRectangle(display,window,gc,*x,CurrentLine*FontHeight,
                       FontWidth,FontHeight);
        XSetForeground(display,gc,gc_values.foreground);
        XSetBackground(display,gc,gc_values.background);

        /* キーシムリターンとタブ（リターンとタブ）が渡されたら、          */
        /* 改行する。                                                      */
        if (keysym == XK_Return || keysym == XK_Tab) {
                CurrentLine = (++CurrentLine)%TextLines;
                *x = 0;
                *y = CurrentLine*FontHeight+FontY;
                XClearArea(display,window,0,CurrentLine*FontHeight,
                           0,FontHeight,False);
        } else {
                int     remain;
                /* 確定文字列を描画する。                                  */
                XmbDrawImageString(display,window,fontset,gc,*x,*y,
                                   string,len);
                /* 新しいスポット・ロケーションを求める。                  */
                *x += XmbTextEscapement(fontset,string,len);

                /* テキスト・バッファの範囲内で確定文字列を格納する。      */
                remain = TextBuffer[CurrentLine].max_size -
                        TextBuffer[CurrentLine].len;
```

```
                if (remain > 0) {
                        TextBuffer[CurrentLine].string =
                                strncat(TextBuffer[CurrentLine].string,
                                        string,MIN(remain,len));
                        TextBuffer[CurrentLine].len =
                                strlen(TextBuffer[CurrentLine].string);
                }

                /* 描画文字列が右端に達したら、改行する。                */
                if (*x > TextCol*FontWidth) {
                        CurrentLine = (++CurrentLine)%TextLines;
                        *x = 0;
                        *y = CurrentLine*FontHeight+FontY;
                        XClearArea(display,window,0,CurrentLine*FontHeight,
                                   0,FontHeight,False);
                }

        }
        /* 新しいカーソルを描画する。                                    */
        XFillRectangle(display,window,gc,*x,CurrentLine*FontHeight,
                       FontWidth,FontHeight);
        return (0);
}

/*
 *      RedisplayText() : テキスト・バッファの内容を再描画する。
 */
static  int
RedisplayText( Display *display, Window window, XFontSet fontset,
               GC gc, XExposeEvent *event )
{
        /* 露出した最初と最後のテキスト・バッファを求める。              */
        int     begin = event->y / FontHeight;
        int     end = (event->y + event->height) / FontHeight;
        int     i, x, y;

        /* 露出したテキスト・バッファだけを再描画する。                  */
        for (i=begin; i<TextLines && i<=end; i++) {
                x = 0;
                y = i * FontHeight + FontY;
                XmbDrawImageString(display,window,fontset,gc,x,y,
                                   TextBuffer[i].string,TextBuffer[i].len);
                /* カーソルの再描画が必要なら、カーソルを描く。          */
                if (i == CurrentLine) {
                        x = XmbTextEscapement(fontset,
                                              TextBuffer[i].string,
                                              TextBuffer[i].len);
                        y = i * FontHeight;
                        XFillRectangle(display,window,gc,x,y,
                                       FontWidth,FontHeight);
                }
        }
        XFlush(display);
}
```

```
/*
 *      usage() : 使用方法を表示する。
 */
static  int
usage( char *argv[] )
{
       fprintf(stderr,"Usage : %s <options>¥n",argv[0]);
        fprintf(stderr,"- Options -¥n");
        fprintf(stderr,"¥t[-lang locale]    specifies locale name¥n");
        fprintf(stderr,"¥t[-mod modifiers]  specifies locale modifiers¥n");
        fprintf(stderr,"¥t[-fs base_name]   specifies fontset name¥n");
        fprintf(stderr,"¥t[-display name]   specifies display name¥n");
        fprintf(stderr,"¥t[-fg color]       specifies foreground color¥n");
        fprintf(stderr,"¥t[-bg color]       specifies background color¥n");
        fprintf(stderr,"¥t[-over]           specifies OverTheSpot style¥n");
        fprintf(stderr,"¥t[-off]            specifies OffTheSpot style¥n");
        fprintf(stderr,"¥t[-root]           specifies Root input style¥n");
        fprintf(stderr,"¥t[-nothing]        specifies Nothing input style¥n");
        fprintf(stderr,"¥t[-none]           specifies None input style¥n");
        fprintf(stderr,"¥n");
        fprintf(stderr,"¥t+-------------+----------+---------+¥n");
        fprintf(stderr,"¥t| Input Style | Preedit  | Status  |¥n");
        fprintf(stderr,"¥t+-------------+----------+---------+¥n");
        fprintf(stderr,"¥t| OverTheSpot | Position | Area    |¥n");
        fprintf(stderr,"¥t| OffTheSpot  | Area     | Area    |¥n");
        fprintf(stderr,"¥t| Root        | Nothing  | Nothing |¥n");
        fprintf(stderr,"¥t| Nothing     | Nothing  | None    |¥n");
        fprintf(stderr,"¥t| None        | None     | None    |¥n");
        fprintf(stderr,"¥t+-------------+----------+---------+¥n");
        fprintf(stderr,"¥n");
        fprintf(stderr,"¥t+--------+-----------+---+---+---+---+---+¥n");
        fprintf(stderr,"¥t|        |           | O | O | R | N | N |¥n");
        fprintf(stderr,"¥t|        |           | v | f | o | o | o |¥n");
        fprintf(stderr,"¥t|        |           | e | f | o | t | n |¥n");
        fprintf(stderr,"¥t| Locale | Modifiers | r |   | t | h | e |¥n");
        fprintf(stderr,"¥t+--------+-----------+---+---+---+---+---+¥n");
        fprintf(stderr,"¥t| C      |           | X | X | X | O | X |¥n");
        fprintf(stderr,"¥t| En_US  |           | X | X | X | O | X |¥n");
        fprintf(stderr,"¥t| en_US  |           | X | X | X | O | X |¥n");
        fprintf(stderr,"¥t| Ja_JP  |           | O | O | O | X | X |¥n");
        fprintf(stderr,"¥t| ja_JP  |           | O | O | O | X | X |¥n");
        fprintf(stderr,"¥t+--------+-----------+---+---+---+---+---+¥n");
}
```

J. プログラム・リスト mrsample.c

```
/*
 * XmRendition サンプル プログラム - mrsample.c
 */
#include <stdio.h>
#include <locale.h>
#include <Xm/XmAll.h>

void btn_CB(Widget w, caddr_t client_data, caddr_t call_data)
{
    exit(0);
}

int main(int argc, char **argv)
{
    XtAppContext  app;
    Widget        top, form, text, btn;
    Arg           args[20];
    int           n;
    XmRendition   rendition[2];
    XmStringTag   renditionTag[] = {"Rendition0", "Rendition1"};
    XmRenderTable renderTable;
    XmString      xms[3];
    Colormap      cmap;
    XColor        color, unused;
    unsigned long pixel_red, pixel_blue;

   /*
    * ロケールを Ja_JP に設定
    */
    setlocale(LC_ALL, "Ja_JP");

    top = XtAppInitialize(&app, "Mrsample", (XrmOptionDescList)NULL, 0,
                          &argc, argv, (String *)NULL, (ArgList)NULL, 0);

    n = 0;
    XtSetArg(args[n], XmNresizePolicy, XmRESIZE_ANY); n++;
    form = XtCreateManagedWidget("Form",
                        xmFormWidgetClass, top, args, n);

   /*
    * テキスト・ウィジェットの作成
    */
    n = 0;
    XtSetArg(args[n], XmNtopOffset, 5); n++;
    XtSetArg(args[n], XmNtopAttachment, XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNleftOffset, 5); n++;
    XtSetArg(args[n], XmNleftAttachment, XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNrightOffset, 5); n++;
    XtSetArg(args[n], XmNrightAttachment, XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNeditMode, XmMULTI_LINE_EDIT); n++;
    XtSetArg(args[n], XmNrows, 8); n++;
    XtSetArg(args[n], XmNcolumns, 20); n++;
    XtSetArg(args[n], XmNvalue, "日本語テキスト(Ja_JP)"); n++;
    text = XtCreateManagedWidget("Text",
                        xmTextWidgetClass, form, args, n);

   /*
    * レンディションを作成しレンダーテーブルに登録
    */
    n = 0;
```

```
    XtSetArg(args[n], XmNfontName, "*-gothic-*--19-*"); n++;
    XtSetArg(args[n], XmNfontType, XmFONT_IS_FONTSET); n++;
    rendition[0] = XmRenditionCreate(text, renditionTag[0], args, n);
    renderTable = XmRenderTableAddRenditions(NULL, rendition, 1,
                                             XmMERGE_REPLACE);
   /*
    * レンダーテーブルをテキスト・ウィジェットに設定
    */
    n = 0;
    XtSetArg(args[n], XmNrenderTable, renderTable); n++;
    XtSetValues(text, args, n);
    XmRenditionFree(rendition[0]);
    XmRenderTableFree(renderTable);

   /*
    * プッシュボタン・ウィジェットの作成
    */
    n = 0;
    XtSetArg(args[n], XmNtopOffset, 5); n++;
    XtSetArg(args[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
    XtSetArg(args[n], XmNtopWidget, text); n++;
    XtSetArg(args[n], XmNleftOffset, 5); n++;
    XtSetArg(args[n], XmNleftAttachment, XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNbottomOffset, 5); n++;
    XtSetArg(args[n], XmNbottomAttachment, XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNmarginHeight, 4); n++;
    XtSetArg(args[n], XmNmarginWidth, 4); n++;
    btn = XtCreateManagedWidget("Btn",
                        xmPushButtonWidgetClass, form, args, n);
    XtAddCallback(btn, XmNactivateCallback, btn_CB, (caddr_t)NULL);

   /*
    * ボタンの文字の色を準備
    */
    XtVaGetValues(btn, XmNcolormap, &cmap, NULL);
    if(XAllocNamedColor(XtDisplay(btn), cmap, "red", &color, &unused)) {
        pixel_red = color.pixel;
    } else {
        pixel_red = XmUNSPECIFIED_PIXEL;
    }
    if(XAllocNamedColor(XtDisplay(btn), cmap, "blue", &color, &unused)) {
        pixel_blue = color.pixel;
    } else {
        pixel_blue = XmUNSPECIFIED_PIXEL;
    }
   /*
    * 二つのレンディションを作成しレンダーテーブルに登録
    */
    n = 0;
    XtSetArg(args[n], XmNrenditionForeground, pixel_red); n++;
    XtSetArg(args[n], XmNfontName, "*-mincho-*--27-*"); n++;
    XtSetArg(args[n], XmNfontType, XmFONT_IS_FONTSET); n++;
    rendition[0] = XmRenditionCreate(btn, renditionTag[0], args, n);
    n = 0;
    XtSetArg(args[n], XmNrenditionForeground, pixel_blue); n++;
    XtSetArg(args[n], XmNfontName, "sans018"); n++;
    XtSetArg(args[n], XmNfontType, XmFONT_IS_FONT); n++;
    XtSetArg(args[n], XmNunderlineType, XmSINGLE_LINE); n++;
    rendition[1] = XmRenditionCreate(btn, renditionTag[1], args, n);
    renderTable = XmRenderTableAddRenditions(NULL, rendition, 2,
                                             XmMERGE_REPLACE);
   /*
    * それぞれのレンディションでボタンの文字を作成
```

```
    */
    xms[0] = XmStringGenerate("終了", NULL, XmCHARSET_TEXT, renditionTag[0]);
    xms[1] = XmStringGenerate("Quit", NULL, XmCHARSET_TEXT, renditionTag[1]);
    xms[2] = XmStringConcat(xms[0], xms[1]);
   /*
    * レンダーテーブルと文字をプッシュボタン・ウィジェットに設定
    */
    n = 0;
    XtSetArg(args[n], XmNrenderTable, renderTable); n++;
    XtSetArg(args[n], XmNlabelString, xms[2]); n++;
    XtSetValues(btn, args, n);
    for(n = 0; n < 2; n++)
        XmRenditionFree(rendition[n]);
    XmRenderTableFree(renderTable);
    for(n = 0; n < 3; n++)
        XmStringFree(xms[n]);

    XtRealizeWidget(top);
    XtAppMainLoop(app);
}
```

K. プログラム・リスト losample.c

```
/*
 * Sample program - losample.c
 * How to compile:
 *    cc -q64 -qlargetype losample.c -o losample  -lXm -lXt -lX11 -li18n
 * How to run:
 *    LANG=JA_JP losample
 */
/*****************/
/* include files */
/*****************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/layout.h>
#include <Xm/Xm.h>
#include <Xm/RowColumn.h>
#include <Xm/PushB.h>
#include <Xm/Text.h>

/***********************/
/* global declarations */
/***********************/
void          Quit(Widget w, caddr_t client_data, caddr_t call_data);
void          Layout(Widget w, caddr_t client_data, caddr_t call_data);
BooleanValue  Shaping(LayoutObject plh);
wchar_t TextValue[] = {0x0e6,0x300,
                       0x254,0x300,0x28c,0x300,0x259,0x300,0x25a,0x300,
                       0x254,0x301,0x28c,0x301,0x259,0x301,0x25a,0x301, 0x0a,
                       0x304b,0x309a,0x304d,0x309a,0x304f,0x309a,0x3051,0x309a,
                       0x3053,0x309a, 0x0a,
                       0x30ab,0x309a,0x30ad,0x309a,0x30af,0x309a,0x30b1,0x309a,
                       0x30b3,0x309a,0x30bb,0x309a,0x30c4,0x309a,0x30c8,0x309a,
                       0x00};

/****************/
/* main program */
/****************/
int main(int argc, char **argv)
{
    /*******************/
    /* local variables */
    /*******************/
    XtAppContext  app;
    Widget        top, rc, quit, layout, text;
    Arg           args[10];
    int           n;

    /*********************************/
    /* set locale (言語環境の確立) */
    /*********************************/
    XtSetLanguageProc(NULL, (XtLanguageProc)NULL, NULL);

    /*************************************************/
    /* initialize toolkit (イントリンシックスの初期化) */
    /*************************************************/
    top = XtAppInitialize(&app, "Losample", (XrmOptionDescList)NULL, 0,
                          &argc, argv, (String *)NULL, (ArgList)NULL, 0);

    /*****************************************/
    /* create widgets (各ウィジェットの生成) */
    /*****************************************/
```

```
    n = 0;
    XtSetArg(args[n], XmNentryAlignment, XmALIGNMENT_CENTER); n++;
    rc = XtCreateManagedWidget("Rc",
                        xmRowColumnWidgetClass, top, args, n);

    n = 0;
    quit = XtCreateManagedWidget("Quit",
                        xmPushButtonWidgetClass, rc, args, n);
    n = 0;
    layout = XtCreateManagedWidget("Layout",
                        xmPushButtonWidgetClass, rc, args, n);

    n = 0;
    XtSetArg(args[n], XmNeditMode, XmMULTI_LINE_EDIT); n++;
    XtSetArg(args[n], XmNrows, 6); n++;
    XtSetArg(args[n], XmNcolumns, 40); n++;
    XtSetArg(args[n], XmNvalueWcs, TextValue); n++;
    text = XtCreateManagedWidget("Text",
                        xmTextWidgetClass, rc, args, n);
    XtAddCallback(quit, XmNactivateCallback, Quit, (caddr_t)NULL);
    XtAddCallback(layout, XmNactivateCallback, Layout, (caddr_t)text);

    /*************************************/
    /* realize widget (ウィジェットの表示) */
    /*************************************/
    XtRealizeWidget(top);
    /*************************************/
    /* main loop (イベント・ループへ入る) */
    /*************************************/
    XtAppMainLoop(app);
}

/*********************/
/* callback routines */
/*********************/
void Quit(Widget w, caddr_t client_data, caddr_t call_data)
{
    exit(0);
}

/*****************************************************************/
/* change text value (レイアウト・サービスによる文字列の変換) */
/*****************************************************************/
void Layout(Widget w, caddr_t client_data, caddr_t call_data)
{
    LayoutObject plh;
    char modifier[40];
    wchar_t *in, *out;
    size_t insize, outsize;
    size_t index = 0;
    int rc;

    bzero(modifier, 40 * sizeof(char));
    plh = m_create_layout(NULL, modifier);
    if(Shaping(plh)) {
        in = XmTextGetStringWcs((Widget)client_data);
        insize = wcslen(in);
        outsize = insize * 2;
        out = (wchar_t *)malloc(outsize * sizeof(wchar_t));
        bzero(out, outsize * sizeof(wchar_t));
        rc = m_wtransform_layout(plh, in, insize, out, &outsize,
                        NULL, NULL, NULL, &index);
        if (rc) {
```

```
            perror("m_wtransform_layout error");
            exit(1);
        }
        XmTextSetStringWcs((Widget)client_data, out);
        XtFree(in);
        free(out);
        m_destroy_layout(plh);
    }
}

BooleanValue Shaping(LayoutObject plh)
{
    LayoutValueRec  lo_values[2];
    BooleanValue    shaping;
    size_t          index = 0;
    int             rc;

    lo_values[0].name = ActiveShapeEditing;
    lo_values[0].value = (LayoutValue) &shaping;
    lo_values[1].name = 0;
    rc = m_getvalues_layout(plh, lo_values, &index);
    if(rc) {
        perror("m_getvalues_layout error");
        exit(1);
    }
    return(shaping);
}
```

特記事項

本書において、日本では発表されていないIBM製品(機械およびプログラム)、プログラミング、またはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのようなIBM製品、プログラミング、またはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書で、IBMライセンス・プログラムまたは他のIBM製品に言及している部分があっても、このことは当該プログラムまたは製品のみが使用可能であることを意味するものではありません。これらのプログラムまたは製品に代えて、IBMの知的所有権を侵害することのない機能的に同等な他社のプログラム、製品、またはサービスを使用することができます。ただし、IBMによって明示的に指定されたものを除き、これらのプログラムまたは製品に関連する稼働の評価および検証はお客様の責任で行っていただきます。

IBMおよび他社は、本書で説明する主題に関する特許権(特許出願を含む)、商標権、または著作権を所有している場合があります。本書は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。実施権、使用権等の許諾については、下記の宛先に、書面にてご照会ください。

〒106-8711
東京都港区六本木 3-2-12
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

商標

本書中、以下の用語は、IBM Corp.の米国およびそのほかの国における商標です。

IBM
AIX
AIXwindows
pSeries

本書中、二重星印（**）の付いている以下の用語は、他社の商標です。

Motif、OSF/Motifは、Open Software Foundation, Incの商標です。

Microsoft、Windowsは、米国Microsoft Corporationの米国および他の国における登録商標です。

Unicodeは、Unicode, Incの商標です。

UNIXは、The Open Groupの米国およびその他の国における登録商標です。

X/Openは、The Open Groupの商標です。

X11、X Window Systemは、Massachusetts Institute of Technologyの商標です

SC88-0428-03

日本アイ・ビー・エム株式会社

〒106-8711　東京都港区六本木 3-2-12