AIX バージョン 7.2

AIX グローバリゼーション

AIX バージョン 7.2

AIX グローバリゼーション

お願い

本書および本書で紹介する製品をご使用になる前に、 253 ページの『特記事項』に記載されている情報をお読みください。

本書は AIX バージョン 7.2 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： AIX Version 7.2
AIX globalization

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

本書は、アプリケーション・プログラマーに対して、アプリケーションで AIX® オペレーティング・システム用の各国語サポートを使用できるようにする方法を詳しく説明します。 さらに、システム管理者に対しては、ネットワーク環境で AIX オペレーティング・システムのグローバリゼーション機能を使用できるようにする方法を詳しく説明します。 プログラマーおよびシステム管理者は、本書を使用して、グローバリゼーションのガイドラインおよび原則についての知識を得ることができます。 本書には、ロケール、コード・セット、入力メソッド、サブルーチン、コンバーター、文字マッピング、国/地域固有の情報、およびメッセージ機能に関するトピックが含まれています。

強調表示

本書では、以下の強調表示規則を使用します。

太字	名前がシステムによって事前定義されているコマンド、サブルーチン、キーワード、ファイル、構造体、ディレクトリー、および他の項目を示します。 さらに、ユーザーが選択するボタン、ラベル、およびアイコンなどのグラフィカル・オブジェクトも示します。
イタリック	実際の名前または値をユーザーが提供する必要があるパラメーターを示します。
モノスペース	具体的なデータ値の例、表示される可能性があるテキストの例、プログラマーとして作成する可能性があるプログラム・コードの一部の例、システムからのメッセージ、またはユーザーが実際に入力する必要がある情報を示します。

AIX での大/小文字の区別

AIX オペレーティング・システムでは、すべてがケース・センシティブとなっています。ケース・センシティブとは、大文字と小文字を区別するという意味です。 例えば、**ls** コマンドを使用するとファイルをリストできます。 `LS` と入力すると、システムはそのコマンドが見つからない (`is not found`) と応答します。 同様に、**FILEA**、**FiLea**、および **filea** は、同じディレクトリーにある場合でも、3 つの異なるファイル名です。 予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

AIX グローバリゼーション

グローバリゼーションには、世界共通のシステムのベースとなる、コマンドと標準的な C ライブラリー・サブルーチンが備えられています。 システムが国際化されているので、以下のような、言語または国/地域ごとに固有な規則に関する標準装備の前提事項または依存関係はありません。

- コード・セット
- 文字種別
- 文字比較規則
- 文字照合順序
- 数字と通貨の形式
- 日付および時刻の形式
- メッセージ・テキスト言語

国/地域別情報や言語に関する情報は、すべてプロセスの実行時に入手します。

国際環境で実行しているシステムを保守するために、グローバリゼーションには以下の機能が備えられています。

- 『プログラムからのメッセージの分離』
- 『コード・セット間の変換』

プログラムからのメッセージの分離

メッセージをプログラムから分けておき、プログラムが実行時にアクセスすることができるメッセージ・カタログの形式で提供する必要があります。 これにより、メッセージをさまざまな言語に翻訳し、その翻訳されたメッセージをユーザーのロケールに基づいてプログラムで使用することが容易になります。

関連概念:

159 ページの『メッセージ機能』
メッセージを、プログラムが実行時にアクセスすることができるメッセージ・カタログの形式で提供し、プログラムから分けておく必要があります。 この調整により、メッセージをさまざまな言語に翻訳し、ユーザーのロケールに基づいてプログラムで使用することが容易になります。 このタスクに役立つコマンドとサブルーチンが、メッセージ機能に備えられています。

コード・セット間の変換

文字とは、データの編成、制御、または表記に使用される記号のことです。特定言語の記述に使用されるこの種の記号のグループにより、文字セットが構成されます。 コード・セットには、文字セットのエンコード値が含まれます。 コード・セット内のエンコード値により、システムとその入出力デバイスとの間のインターフェースが設けられます。各国語サポートには、さまざまなコード・セット内にある文字エンコード値に従うコンバーターが備わっています。

かつては英字をエンコードする作業が主に行われてきました。英字の字数は多くないので、この場合 7 ビットのエンコード・メソッドを使用すれば十分でした。 中国語、日本語、韓国語などのアジア言語のように、さらに多数の字数をサポートするために、マルチバイト・エンコードを含む追加のコード・セットが開発されました。現在、世界規模の情報処理をサポートする文字セットである Unicode が、オペレーティン

グ・システム・レベルでの基本交換フォーマットとして使用されています。UTF-8、UTF-16、および UTF-32 コード・セットが、システム・アプリケーション用の主な Unicode エンコード・スキームです。

国際化されたプログラムは、異なるコード・セット環境で生成されたデータを正確に読み取り、情報を正確に処理する必要があります。現行コード・セットを知っていると、コード・セット変換に役立てることができます。**nl_langinfo(CODESET)** サブルーチンを使用して、処理中の現行コード・セットを入手することができます。この戻り値は `char` ポインターで、これはシステム中のコード・セットの名前です。

関連概念:

85 ページの『プログラミングのためのコンバーターの概要』
各国語サポートには、データを (多くの場合) コード・セット間で変更できるグローバリゼーションのベースが備えられています。この目的のために、いくつかの標準コンバーターがサポートされています。

入力メソッドのサポート

文字セットが大規模な言語の場合は、文字の入力は複雑になります。 例えば、中国語、韓国語、および日本語は字数が多いので、文字に対するキー・ストロークについて、 1 対 1 のキー・マッピングを備えられません。しかしながら、特殊な入力メソッドを使用すると、ユーザーは文字を音声入力したりキー入力したりでき、それらの文字を固有の言語の文字に変換できます。

個々のキーボードに関連したキーボード・マップにより、 1 つ以上のキー・ストロークが適切な文字エンコードに突き合わせられます。

関連概念:

126 ページの『入力メソッド』
グローバリゼーションがベースを提供する国際環境で実行するアプリケーションの場合は、入力メソッドが必要です。入力メソッドは、アプリケーション・プログラミング・インターフェース (API) であり、特定の言語、キーボード、またはコード・セットに依存しないアプリケーションを開発できるようにします。

コンバーターの概要

グローバリゼーションには、データをコード・セット間で変更できるグローバリゼーションのベースが備えられています。テキスト・ファイルやメッセージ・カタログを変換する必要が生じることがあります。 この種の変換用に、複数の標準的なコンバーターがあります。

あるプログラムからリモート・ホスト上の別のプログラムにデータを送信する際に、送信元のマシンのコード・セットから受信側のコード・セットにデータを変換する必要が生じることがあります。 例えば、IBM® VM システムと通信すると、このシステムは ISO-8859-1 データを EBCDIC に変換します。 コード・セットは、文字を定義し、コード・ポイントへの機能割り当てを制御します。 プログラムが受信したコード・セットとは違うコード・セットでデータを表示する際には、これらのコード化文字を変換しなければなりません。

関連概念:

85 ページの『プログラミングのためのコンバーターの概要』
各国語サポートには、データを (多くの場合) コード・セット間で変更できるグローバリゼーションのベースが備えられています。この目的のために、いくつかの標準コンバーターがサポートされています。

メッセージ機能の使用

メッセージを容易にさまざまな言語に翻訳できるようにしたり、それらのメッセージをユーザーのロケールを基にしたプログラムで使用できるようにしたりするには、メッセージをプログラムから分離した状態を保持し、プログラムの実行時にアクセスできるメッセージ・カタログの形式で備える必要があります。 メッセージ機能には、このタスクに役立つコマンドとサブルーチンが備えられています。 プログラマーはアプ

リケーション・メッセージを含むメッセージ送信元ファイルを作成し、これらのファイルはメッセージ・カタログに変換されます。これらのカタログはアプリケーションで使用され、必要に応じてメッセージの検索や表示が行われます。 プログラムの変更や再コンパイルを行わずに、メッセージ送信元ファイルを他の言語に翻訳したり、メッセージ・カタログに変換したりできます。

メッセージ機能には以下のコマンドが組み込まれており、これらのコマンドはシェル・スクリプトを使用してメッセージを表示したり、コマンド・ラインからメッセージを表示したりします。

コマンド	説明
dspcat	メッセージ・カタログのすべてまたは一部を表示します。
dspmsg	メッセージ・カタログから選択したメッセージを表示します。

これらのコマンドは、NLSPATH 環境変数を使用して、指定されたメッセージ・カタログを見付けます。NLSPATH 環境変数は、メッセージ・カタログを含むディレクトリーをリストします。 これらのディレクトリーは、リストされている順序で検索されます。 以下に例を示します。

```
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/prime/%N
```

%L および %N 特殊変数は、以下のように定義されています。

特殊変数	説明
%L	メッセージ・カタログを含むロケール固有のディレクトリーを指定します。 **LC_MESSAGES** カテゴリーまたは **LANG** 環境変数の値が、ディレクトリー名に使用されます。ユーザーは、**LANG**、**LC_ALL**、または **LC_MESSAGES** 環境変数を、メッセージ・カタログのロケールに設定できます。
%N	オープンするカタログの名前を指定します。

dspcat コマンドがメッセージを検索できない場合は、デフォルトのメッセージが表示されます。 デフォルトのメッセージに **%*n*$** という形式のストリングが含まれている場合は、そのデフォルトのメッセージを単一引用符で囲まなければなりません。 **dspcat** コマンドがメッセージを検索できない場合に、デフォルトのメッセージを指定していないと、システム生成のエラー・メッセージが表示されます。

以下の例では、**dspcat** コマンドを使用して、既存の msgerrs.cat メッセージ・カタログ中のメッセージをすべて表示します。

```
/usr/lib/nls/msg/$LANG/msgerrs.cat:
dspcat msgerrs.cat
```

以下の出力が表示されます。

```
1:1 Cannot open message catalog %s
Maximum number of catalogs already open
1:2 File %s not executable
2:1 Message %d, Set %d not found
```

この方法でメッセージ・カタログの内容を表示し、**mkcatdefs** コマンドを使用してシンボリック ID を交換することにより、 msgerrs メッセージ送信元ファイルに割り当てられたメッセージ ID 番号を検索できます。 シンボリック ID を **dspmsg** コマンドの参照として使用することは容易ではありませんが、上記のように **dspcat** コマンドを使用して必要な ID 番号を取得できます。

以下は **runtest** という単純なシェル・スクリプトで、**dspmsg** コマンドの使用方法を示します。

```
if [ - x ./test ]
    ./test;
else
    dspmsg  msgerrs.cat -s 1 2 '%s NOT EXECUTABLE ¥n' "test";
    exit;
```

注**:** 上記の例のように、絶対パス名を使用しない場合は、**NLSPATH** 環境変数を設定する際に、**dspcat** コマンドが正しいカタログのディレクトリーを検索できるように注意してください。 **LC_MESSAGES** カテゴリーや **LANG** 環境変数の値は、ディレクトリー検索パスにも影響します。

デバイス用の各国語サポートの設定

各国語サポートではロケール設定を使用して環境を定義します。 ロケール設定は、ユーザーのデータ処理と言語の要件に依存し、これにより入出力デバイスの要件が決定されます。 ユーザーのロケールに従ったデバイスの構成は、システム管理者が担当します。

端末装置 (tty デバイス)

setmaps コマンドを使用して、特定の tty や pty の端末装置とコード・セットのマップを設定してください。 **setmaps** ファイル形式は、コード・セット・マップ・ファイルと端末装置マップ・ファイルを定義します。

コード・セット・マップ・ファイルのテキストは、コード・セットの説明であり、タイプ (単一バイトまたはマルチバイト)、メモリーと画面の幅 (マルチバイト・コード・セットの場合)、およびストリーム上にプッシュするオプションのコンバーター・モジュールが含まれます。 コード・セット・マップ・ファイルは、 **/usr/lib/nls/csmap** ディレクトリーにあり、コード・セットと同じ名前です。 詳しくは、「プログラミングの一般概念: プログラムの作成およびデバッグ」の『コンバーター・モジュール』を参照してください。

プリンター

仮想プリンターは、着信ジョブのデフォルトのコード・セットを、 **/etc/environment** ファイル中の **LANG** エントリーから継承します。プリンター・サブシステムは複数の仮想プリンターをサポートできます。 複数の仮想プリンターをサポートしている場合、その個々のコード・セットは違っていてもかまいません。 以下の分野のプリンター・サブシステムのシナリオが推奨されています。

- 1 つ目のシナリオには、複数のキュー、複数の仮想プリンター、および 1 つの物理プリンターが関係します。 仮想プリンターごとに独自のコード・セットがあります。印刷コマンドは、使用するキューを指定します。 さらにキューは、適切なコード・セットのある仮想プリンターを指定します。 このシナリオの場合、ユーザーはどのキューがどの仮想プリンターに接続しているか知っている必要があり、また個々の仮想プリンターに関連しているコード・セットを知っている必要があります。
- 2 つ目のシナリオは 1 つ目に似ていますが、個々の仮想プリンターが別のプリンターに接続しています。
- 3 つ目のシナリオでは、**qprt** コマンドを使用してコード・セットを指定します。 このオプションの場合は、複数のキューが使用でき、1 つの仮想プリンターがあります。 仮想プリンターは、継承されたデフォルトのコード・セットを使用します。

-P-x フラグを指定して **qprt** コマンドを使用し、キューとコード・セットを指定してください。 **-P** フラグを指定しないと、デフォルト・キューが使用されます。 **-x** フラグを使用しないと、仮想プリンターのデフォルト・コード・セットが使用されます。

低機能端末 (LFT)

LFT は、キー・マップを使用して単一バイト・コード・セット言語をサポートしています。 LFT キー・マップは、キー・ストロークをコード・セット中の文字ストリングに変換します。 使用できるすべてのキー・マップのリストが、 **/usr/lib/nls/loc** ディレクトリー中にあります。 LFT では、マルチバイト・コード・セットが必要な言語はサポートされていません。

デフォルトの LFT キーボード設定と、関連したフォント設定は、インストール時に選択した言語に基づいています。 以下のデフォルト・コード・セットを使用できます。

- ISO8859-1
- ISO8859-2
- ISO8859-4
- ISO8859-5
- ISO8859-6
- ISO8859-7
- ISO8859-8
- ISO8859-9
- ISO8859-15

以下の方法で、デフォルト設定を変更できます。

- 次のリブート時のデフォルト・フォントを変更するには、**-n** フラグを指定して **chfont** コマンドを使用します。
- 次のリブート時のデフォルト・キーボードを変更するには、**-n** フラグを指定して **chkbd** コマンドを使用します。

lsfont コマンドと **lskbd** コマンドは、現在 LFT で使用できるフォントとキーボードのマップをすべてリストします。

サポートされているすべてのコード・セットに関する LFT フォント・ライブラリーは、**/usr/lpp/fonts** ディレクトリー中にあります。

言語環境の変更

多数のシステム操作が言語環境の影響を受けます。 この種の操作には、照合、時刻と日付の表現、数値表現、通貨の表現、およびメッセージ変換が含まれます。 言語環境は **LANG** 環境変数の値によって決まり、 **chlang** コマンドを使用してこの値を変更できます。 **chlang** コマンドは、コマンド・ラインまたは SMIT から実行できます。

SMIT 高速パスを使用して言語環境を変更するには、コマンド・ラインで `smit chlang` と入力してください。

デフォルト・キーボード・マップの変更

使用したい言語の正しいキーボードを指定できます。オペレーティング・システムには、そのための多数のキーボード・マップが備えられています。 LFT 端末のデフォルトのキーボード・マップは、SMIT 高速パスの **smit chkbd** コマンドまたは **chkbd** コマンドを使用して、変更できます。システムを再始動するまで変更内容は有効になりません。

注**:** 特定の物理キーボードが使用中であると想定しないでください。ロケール設定に基づく入力メソッドを使用してキーボード入力を処理してください。

ICU4C ライブラリー

共通コンポーネントである ICU4C は、International Components for Unicode for C/C++ クラス・ライブラリーの略です。これは、C/C++ プログラム言語でグローバルなアプリケーションを作成するためのグ

ローバリゼーション・ユーティリティーを提供します。 ICU4C ライブラリーは、AIX オペレーティング・システム上で実行される多数の製品によって使用されています。

ロケール

国際化されたシステムには、コード・セット、文字種別、文字比較規則、文字照合順序、通貨の形式設定、数字の句読点、日付および時刻の形式設定、またはメッセージのテキストに関する標準装備の前提事項または依存関係はありません。これらの言語や国/地域別情報によってロケール が定義されます。 国際化されたシステムは、さまざまな場所の情報を正しく処理します。例えば、米国の日付形式 9/6/2015 は、2015 年 9 月 6 日と解釈されます。英国では同じ日付形式が、2015 年の 6 月 9 日と解釈されます。数字や通貨のデータの形式設定も国特有です。例えば、米国のドルや英国のポンドなどがあります。

すべてのロケール情報はプログラムの実行時にアクセス可能になっており、ご使用の国/地域別情報や言語でデータを正しく処理したり表示したりできるようになっていなければなりません。 このプロセスのことを各国語サポートといいます。各国語サポートは、データの形式設定に関するロケール固有の規則を含むデータベースの作成とその規則を入手するためのインターフェースの開発から成ります。

ロケールについて

ロケールは、言語規則のセットの識別に使用する言語、地域、およびコード・セットの組み合わせから成ります。 この種の規則には、照合、大/小文字の変換、文字種別、メッセージ・カタログの言語、日付および時刻の表現、通貨記号、および数字表現に関する情報が含まれます。

最初に **localedef** コマンドによって、ロケール定義送信元ファイルに含まれているロケール情報を、ロケール・データベースに変換しなければなりません。変更後、**setlocale** サブルーチンが、この情報にアクセスしてアプリケーション用にロケール情報を設定できるようになります。 論理的な方法でロケール・データを処理するために、ロケール定義送信元ファイルは 6 つのカテゴリーに分かれています。 個々のカテゴリーにはロケール・データの特定の局面が含まれています。 **LC_*** 環境変数と LANG 環境変数を使用して、ご希望のロケールを指定できます。

関連概念:

8 ページの『ロケール・カテゴリーについて』

ロケール・カテゴリー とは、言語および国/地域別情報ごとに固有のデータの、特定のグループのことです。 例えば、日付および時刻の形式、曜日の名前、月の名前、およびその他の時間に関連する固有情報を参照するデータは、**LC_TIME** カテゴリーにグループ化されます。 個々のカテゴリーは、ロケール・サブセットの事項について記述したキーワードのセットを使用します。

標準的なユーザーのシナリオ

システム上で複数の各国語シナリオを検出する場合があります。このセクションでは、一般的なシナリオと、推奨アクションをリストします。

- ユーザーがデフォルトのコード・セットを保持する。

  複数のコード・セットがサポートされている場合でも、言語と地域の組み合わせについては、デフォルトのコード・セットで十分に対応できる場合があります。現行ユーザー環境でデフォルトのコード・セットが使用されている場合や、ユーザーが初心者でコード・セット設定がない場合は、デフォルトのコード・セットを保持する場合があります。

  システムのインストール時に選択される言語と地域が、デフォルトのコード・セットに基づく適切なロケールのデフォルトになります。デフォルトのキーボード・マッピング、デフォルトのフォント、およ

びメッセージ・カタログはすべてデフォルトのコード・セットに基づいて確立されます。 このシナリオには、ユーザーからの特別なアクションは不要です。

- ユーザーがデフォルトのコード・セットからコード・セットを変更する。

  Latin 1 または日本語のロケールのユーザーが、データと各国語サポート環境を別の (デフォルトでない) コード・セットに移行したい場合があります。 この移行は以下の方法で行うことができます。

  – 変換を必要とする既存のデータがある場合。

    優先コード・セットに変換する必要のあるフラット・テキスト・ファイルは、SMIT の「**Manage the Language Environment (**言語環境の管理**)**」メニュー、または **iconv** ユーティリティーを使用して変換できます。ユーザー定義の構造化ファイルは、構造化ファイル内の必要なテキスト・フィールドを変換するために **iconv** ライブラリー機能を使用するユーザー作成の変換ツールで変換する必要があります。

  – 他のコード・セットに変更したい場合。

    言語と地域の組み合わせについて複数のコード・セットがサポートされている場合、以下のオプションを使用して、デフォルト以外のロケールに変更できます。

    - SMIT の「**Manage the Language Environment (**言語環境の管理**)**」メニュー。
    - **chlang**、**chkbd**、および **chfont** コマンド

ロケールの命名規則

個々のロケールの名前は、ロケール定義送信元ファイルの名前になります。 この種のファイルには、記述されている言語、地域、およびコード・セットの情報の名前が付けられます。 ロケール定義ファイルの命名には、以下の形式が使用されます。

```
language[_territory][.codeset][@modifier]
```

例えば、ISO8859-1 コード・セットを使用した、デンマークで話されているデンマーク語のロケールは、`da_DK.ISO8859-1` です。 `da` はデンマーク語を表し、 `DK` はデンマークを表します。 簡略形式の `da_DK` でもこのロケールを示せます。 ISO8859–15 コード・セットを使用している同じ言語と地域は、`da_DK.8859–15` と示されます。

システム定義のロケール定義ファイルが備えられており、ロケールのカテゴリーとキーワードの形式が示されています。 システム定義ロケールのロケール定義ファイルは、 **/usr/lib/nls/loc** ディレクトリーに含まれています。 C または POSIX ロケールは、起動時にすべてのプロセスに継承される ANSI C 定義の標準ロケールを定義します。 システム定義のロケール定義送信元ファイルのリストを入手するには、コマンド・ラインで以下のように入力してください。

```
/usr/lib/nls/lsmle -c
```

インストール・デフォルト・ロケール

インストール・デフォルト・ロケールとは、インストール時に選択したロケールのことです。 例えば、プロンプトが出されたら、ユーザーはインストール・プロセス時にカナダで話されているフランス語を指定できます。コード・セットのデフォルトは自動的に ISO8859-1 コード・セットになります。

システムでは、この情報を使用して、**LANG** 環境変数によって指定されたデフォルト・ロケールの値を、**fr_CA** (**fr** は ISO8859-1 フランス語、**CA** はカナダ) に設定します。 **LC_*** または **LANG** 環境変数を変更しない限り、すべてのプロセスでこのロケールが使用されます。 SMIT 中の「言語環境の管理」メニューを使用して、デフォルト・ロケールを変更できます。

システム・バックアップが作成されて再インストールされる場合、デフォルト・ロケール値は、/bosinst.data ファイルと、使用可能であれば、/var/adm/ras/bosinst.data ファイルで使用されます。**smit mlang** コマンドを使用してロケール値を変更した場合、これら 2 つのファイルは自動的に更新されません。このシナリオでは、実行中のシステムのロケール値を一致させるために、/bosinst.data ファイルと、使用可能であれば、/var/adm/ras/bosinst.data ファイル内のスタンザを変更する必要があります。

関連情報:

使用可能なシステム管理インターフェース

C または POSIX のロケール

このロケールは、起動時にすべてのプロセスで継承されるロケールの、ANSI C または POSIX 定義の標準です。 C または POSIX ロケールは、7 ビットの ASCII 文字セットが前提になっており、前述の 6 つのカテゴリーを定義します。

ロケール・カテゴリーについて

ロケール・カテゴリー とは、言語および国/地域別情報ごとに固有のデータの、特定のグループのことです。 例えば、日付および時刻の形式、曜日の名前、月の名前、およびその他の時間に関連する固有情報を参照するデータは、**LC_TIME** カテゴリーにグループ化されます。 個々のカテゴリーは、ロケール・サブセットの事項について記述したキーワードのセットを使用します。

ロケール定義送信元ファイルに、以下の標準カテゴリーを定義できます。

LC_COLLATE
文字やストリングの照合に関する情報を定義します。

LC_CTYPE
文字種別、大/小文字の変換、およびその他の文字属性を定義します。

LC_MESSAGES
肯定応答と否定応答の形式を定義します。

LC_MONETARY
通貨の数字情報を形式設定する規則とシンボルを定義します。

LC_NUMERIC
通貨以外の数字情報を形式設定する規則とシンボルを定義します。

LC_TIME
時刻と日付の情報を形式設定する規則とシンボルのリストを定義します。

注**:** ロケール・カテゴリーを変更する方法は、ロケール定義送信元ファイルを編集することのみに限られます。 同じ名前の環境変数と混同しないでください。この変数はコマンド・ラインから設定できます。

関連概念:

6 ページの『ロケールについて』
ロケールは、言語規則のセットの識別に使用する言語、地域、およびコード・セットの組み合わせから成ります。 この種の規則には、照合、大/小文字の変換、文字種別、メッセージ・カタログの言語、日付および時刻の表現、通貨記号、および数字表現に関する情報が含まれます。

ロケール環境変数について

各国語サポートは、複数の環境変数を使用してロケールの選択を制御します。 これらの変数の値を設定して、ロケール情報の検索パスを変更できます。

LANG
インストール・デフォルト・ロケールを指定します。

注**: LANG** 環境変数の値は、インストール時に確立されます (この値は、**LC_*** 環境変数を設定しない場合にすべてのプロセスで使用されるロケールです)。 SMIT 中の「Manage Language Environment (言語環境の管理)」メニューを使用して、 **LANG** 環境変数を変更できます。SMIT の使用方法について詳しくは、オペレーティング・システムおよびデバイスの管理 中の『使用可能な system management interfaces (SMIT)』を参照してください。 C ロケールと POSIX ロケールは、最高のパフォーマンスを実現するように設計されています。

LC_ALL
LANG 環境変数の値や、その他の **LC_*** 環境変数の値を指定変更します。

LC_COLLATE
LC_COLLATE カテゴリー情報に使用するロケールを指定します。 **LC_COLLATE** カテゴリーにより、範囲、等価クラス、および複数文字の照合エレメントの動作を管理する文字照合やストリング照合の規則が決まります。

LC_CTYPE
LC_CTYPE カテゴリー情報に使用するロケールを指定します。 **LC_CTYPE** カテゴリーにより、テキスト・データ文字のバイトのシーケンスの解釈 (単一バイト文字とマルチバイト文字のどちらか)、文字の種別 (アルファベットや数字など)、および文字クラスの動作を管理する文字処理規則が決まります。

LC__FASTMSG
C ロケールと POSIX ロケールにデフォルト・メッセージを使用することと、 **LC__FASTMSG** を**真**に設定した場合に `NLSPATH` を無視することを指定します。 指定しないと、POSIX に準拠したメッセージ処理が実行されます。 デフォルト値は、 **/etc/environment** ファイル中の `LC__FASTMSG=true` になります。

LC_MESSAGES
LC_MESSAGES カテゴリー情報に使用するロケールを指定します。 **LC_MESSAGES** カテゴリーにより、肯定および否定応答と、メッセージやメニューのロケール (言語) を管理する規則が決まります。

端末でマルチバイト文字を表示しないアプリケーションを作成するアプリケーション開発者は、**LC_MESSAGES** 値が `C@lft` に設定されていないことを確認してください。 必要ならば、**putenv("LC_MESSAGES=")** サブルーチンでその設定を使用不可にします。 その結果、翻訳されたメッセージ・カタログを使用する出力が出ます。 `C@lft` は、マルチバイト文字を表示できるログイン・セッションによって使用不可にされます。 **cron** または **inittab** を使用して起動されたプロセスは、`C@lft` **LC_MESSAGES** 値を保存し、**setlocale()** サブルーチンを使用して、デフォルト・メッセージのための言語環境を設定します。

LC_MONETARY
LC_MONETARY カテゴリー情報に使用するロケールを指定します。 **LC_MONETARY** カテゴリーにより、通貨に関連した形式設定を管理する規則が決まります。

LC_NUMERIC
LC_NUMERIC カテゴリー情報に使用するロケールを指定します。 **LC_NUMERIC** カテゴリーにより、通貨以外の数字の形式設定を管理する規則が決まります。

LC_TIME
LC_TIME カテゴリー情報に使用するロケールを指定します。 **LC_TIME** カテゴリーにより、日付および時刻の形式設定を管理する規則が決まります。

LOCPATH
バイナリー・ロケール・ファイル、入力メソッド、コード・セット・コンバーターを含むローカライズされた情報の検索パスを指定します。

注**:** すべての **setuid** プログラムと **setgid** プログラムは、 **LOCPATH** 環境変数を無視します。

NLSPATH
メッセージ・カタログ・ファイルを検索する際の検索パスを指定します。この環境変数は、各国語サポート・サブシステムのメッセージ機能コンポーネントで使用されます。 **NLSPATH** 変数の必要な形式についての詳細は、 **catopen** サブルーチンを参照してください。

ロケールの選択に影響する環境変数を、高、中、および低の 3 つの優先順位クラスにグループ分けできます。 「高」優先順位クラスの環境変数は以下のとおりです。

- **LC_ALL**
- **LC_COLLATE**
- **LC_CTYPE**

「中」優先順位クラスの環境変数は以下のとおりです。

- **LC_MESSAGES**
- **LC_MONETARY**
- **LC_NUMERIC**
- **LC_TIME**

「低」優先順位クラスの環境変数は以下のとおりです。

- **LANG**

setlocale サブルーチンにより、特定のカテゴリーかすべてのカテゴリーのロケールが要求されると、以下のように環境変数の設定がそれぞれの優先順位レベルによって照会されます。

- **LC_ALL** 環境変数が設定されている場合は、この変数で指定されているロケールが 6 つのカテゴリーすべてで使用される。 例えば、 **LC_ALL** 環境変数の値が `en_US` で、 **LANG** 環境変数の値が `fr_FR` の場合は、 **setlocale** サブルーチンを呼び出すと、 6 つのカテゴリーがそれぞれ `en_US` ロケールに設定されます。
- **LC_ALL** 環境変数が設定されていない場合は、カテゴリーごとに対応する環境変数によって指定されているロケールが使用される。 例えば、**LC_ALL** 環境変数が設定されておらず、 **LC_COLLATE** 環境変数が `de_DE` に設定されており、 **LC_TIME** 環境変数が `fr_CA` に設定されている場合は、 **setlocale** サブルーチンを呼び出すと、 **LC_COLLATE** カテゴリーは `de_DE` に設定され、 **LC_TIME** カテゴリーは `fr_CA` に設定されます。 この状態の場合、どの環境変数も他の環境変数に優先しません。
- **LC_ALL** 環境変数が設定されておらず、特定の **LC_*** 環境変数の値が設定されていない場合は、**LANG** 環境変数の値によって、その特定のカテゴリーの設定が決まる。 例えば、**LC_ALL** 環境変数が設定されておらず、 **LC_CTYPE** 環境変数が `en_US` に設定されており、 **LC_NUMERIC** 環境変数が設定されておらず、 **LANG** 環境変数が `is_IS` に設定されている場合は、 **setlocale** サブルーチンを呼び出すと、 **LC_CTYPE** カテゴリーは `en_US` に設定され、 **LC_NUMERIC** カテゴリーは `is_IS` に設定されます。 **LANG** 環境変数は、`LC_*` 環境変数によってあらかじめ決められていないカテゴリーのみのロケールを指定します。
- **LC_ALL** 環境変数が設定されておらず、特定の **LC_*** 環境変数の値が設定されておらず、 **LANG** 環境変数の値が設定されていない場合は、その特定のカテゴリーのロケールはデフォルトの `C` ロケールになる。例えば、**LC_ALL** 環境変数が設定されておらず、 **LC_MONETARY** 環境変数が `sv_SE` に設定さ

れており、 **LC_TIME** 環境変数が設定されておらず、 **LANG** 環境変数が設定されていない場合は、**setlocale** サブルーチンを呼び出すと、 **LC_MONETARY** カテゴリーは `sv_SE` に設定され、**LC_TIME** カテゴリーは `C` に設定されます。

環境変数の優先順位の例

以下の表は、環境変数の現行設定と、**setlocale(LC_ALL,"")** 呼び出しの影響を示しています。最後の列は、**setlocale(LC_ALL,"")** の呼び出し後のロケール設定を示しています。

表 *1.* 環境変数の優先順位の例

環境変数およびカテゴリー名	環境変数の値	**setlocale(LC_ALL,"")** 呼び出し後のカテゴリーの値
LC_COLLATE	de_DE	de_DE
LC_CTYPE	de_DE	de_DE
LC_MONETARY	en_US	en_US
LC_NUMERIC	(設定なし)	da_DK
LC_TIME	(設定なし)	da_DK
LC_MESSAGES	(設定なし)	da_DK
LC_ALL	(設定なし)	(適用なし)
LANG	da_DK	(適用なし)

ロケール定義送信元ファイルについて

コマンド・ラインから設定できる環境変数とは違って、ロケールを変更するには、ロケール定義送信元ファイルを編集してコンパイルするしかありません。

ご希望のロケールがライブラリー中に存在しない場合は、 **localedef** コマンドによって、バイナリー・バージョンのロケールをコンパイルできます。プログラムのロケールに関する動作がロケール定義送信元ファイルに制御されるようにするには、まず **localedef** コマンドによってこのファイルを変換し、プログラムがロケール・オブジェクトを使用できるようにしなければなりません。 **localedef** コマンドは、ロケールの定義を含む送信元ファイルを実行時形式に変換し、コマンド・ラインで指定したファイル (普通はロケール名) に実行時バージョンをコピーします。 その後、国際化されたコマンドやサブルーチンがロケール情報にアクセスできるようになります。 **localedef** コマンドによって送信元ファイルを変換する準備に関する情報は、ファイル参照 の『**Locale Definition Source File Format**』を参照してください。

マルチバイト・サブルーチン

マルチバイト・サブルーチンは、ファイル・コード形式の文字を処理します。普通この種のサブルーチンの名前の先頭は、接頭部 **mb** になります。しかしながら、この接頭部が存在しないマルチバイト・サブルーチンもあります。 例えば、 **strcoll** サブルーチンと **strxfrm** サブルーチンは、マルチバイト形式の文字を処理しますが、 **mb** 接頭部はありません。標準的な C サブルーチン **strcmp**、 **strcpy**、**strncmp**、**strncpy**、**strcat**、および **strncat** は、バイトに対して操作を行い、マルチバイト・データの処理に使用できます。 標準的な C 検索サブルーチン **strchr**、 **strrchr**、**strpbrk**、 **strcspn**、**strrchr**、 **strspn**、**strstr**、および **strtok** は、以下の場合に使用できます。

- 単一バイト・コード・セット中の文字の検索やスキャン
- マルチバイト・ストリング中の固有のコード・ポイント範囲の文字の検索やスキャン

関連概念:

14 ページの『各国語サポートのサブルーチン』
このセクションでは、プログラマーが国際化されたポータブル・プログラムを開発する際にサブルーチンを使用する場合の指針を示します。可搬性を最大限にするには、標準的な Open Group、ISO/ANSI C、お

よび POSIX 関数を使用してください。

ワイド文字サブルーチン

ワイド文字サブルーチンは、プロセス・コード形式の文字を処理します。 普通、ワイド文字サブルーチンの先頭は、接頭部 **wc** になります。しかしながら、この規則には例外があります。 例えば、ワイド文字種別関数は **isw** 接頭部を使用します。 特定のサブルーチンがワイド文字サブルーチンかどうかを判断するには、サブルーチンのプロトタイプで文字が **wchar_t** データ型または **wchar_t** データ・ポインターとして定義されているか、またはサブルーチンが **wchar_t** データ型を戻すかどうか検査してください。 この規則には例外があります。 例えば、ワイド文字種別サブルーチンは **wint_t** データ型値を受け入れます。

関連概念:

14 ページの『各国語サポートのサブルーチン』
このセクションでは、プログラマーが国際化されたポータブル・プログラムを開発する際にサブルーチンを使用する場合の指針を示します。可搬性を最大限にするには、標準的な Open Group、ISO/ANSI C、および POSIX 関数を使用してください。

両方向性と文字成形

国際化されたプログラムが、テキストの両方向性や字形を処理する必要が生じることがあります。

両方向性 (BIDI) は、読む方向の異なるテキストが共存している場合に該当します。 例えば、英語のテキストは左から右に読みます。 ヘブライ語のテキストは右から左に読みます。 同じ行に英語とヘブライ語のテキストが現れれば、このテキストは両方向です。

字形 は、文字の形状が、テキスト行におけるその位置によって決まるときに発生します。 アラビア語のように、言語によっては、文字の形状がストリング内のその位置や、周囲の文字によって変わるものもあります。

関連概念:

176 ページの『レイアウト (両方向テキストと文字成形) の概要』
両方向 (BIDI) テキストが発生するのは、方向の異なるテキストが一緒に現れる場合です。 例えば、英語のテキストは左から右に読みます。 アラビア語とヘブライ語のテキストは右から左に読みます。 同じ行に英語とヘブライ語のテキストが現れれば、このテキストは両方向です。

コード・セットからの独立性

システムが外部の環境と通信するには、コード・セットに関する特定の情報が必要です。 この情報は、コード・セットから独立したライブラリー・サブルーチン (グローバリゼーション・ライブラリー) によって隠されています。この種のサブルーチンは、コード・セットに従属する関数に情報を渡します。各国語サポート・サブルーチンが必要なコード・セット情報を処理するため、コード・セットに関する十分な知識がなくても、文字を処理するプログラムを作成できます。 このようなプログラミング手法のことを、コード・セットの独立性 といいます。

関連概念:

243 ページの『各国語サポートのサンプル・プログラム』
このセクションには、サンプル・プログラムの一部である `my_example.c` が含まれています。これは、コード・セットから独立したプログラミングによるグローバリゼーションを示しています。

コード・セット内の最大バイト数の決定

MB_CUR_MAX マクロを使用して、現行ロケール中のコード・セットのマルチバイト文字中の最大バイト数を決定できます。 このマクロの値は、 **LC_CTYPE** カテゴリーの現行設定に応じて変わります。 ロケ

ールはプロセス間で違う場合があるので、さまざまなプロセスやさまざまな時刻に **MB_CUR_MAX** マクロを実行すると、結果が異なることがあります。 **MB_CUR_MAX** マクロは、 **stdlib.h** ヘッダー・ファイル中に定義されます。

MB_LEN_MAX マクロを使用して、システムでサポートされているコード・セット中の最大バイト数を決定できます。このマクロは、 **limits.h** ヘッダー・ファイル中に定義されます。

文字とストリングの表示幅の決定

_max_disp_width マクロは、オペレーティング・システムごとに固有なので、ポータブル・アプリケーションでは使用しないようにする必要があります。 可搬性が重要でない場合は、**_max_disp_width** マクロを使用して、現行ロケール中のコード・セット中の単一文字にとって必要な表示列の最大数を決定できます。 このマクロの値は、 **LC_CTYPE** カテゴリーの現行設定に応じて変わります。この値が 1 の場合は、現行コード・セット中のすべての文字について、出力上に 1 列の表示列が必要です。

MB_CUR_MAX と **_max_disp_width** が両方とも 1 に設定されている場合は、 **strlen** サブルーチンを使用して、ストリングにとって必要な表示列の幅を決定できます。 **MB_CUR_MAX** が 1 より大きい場合は、 **wcswidth** サブルーチンを使用して、ストリングの表示列の幅を検索してください。

wcswidth と **wcwidth** のワイド文字表示幅サブルーチンには、対応するマルチバイト関数はありません。**wcswidth** サブルーチンは、ディスプレイ上の使用可能なスペースに表示できる文字数は示しません。 この場合は、**wcwidth** サブルーチンが役立ちます。 ワイド文字のストリングに対してこのサブルーチンを繰り返し呼び出して、ディスプレイ上の使用可能な場所に表示できる文字数を検索できます。

コード・セットの認識に対する例外: 固有のコード・ポイント範囲

サポートされているコード・セットの編成方法に基づいて、「プログラム中で基礎的なコード・セットを認識していることは想定されない」という前提に対する例外が 1 つあります。

マルチバイト文字のストリング中で、固有のコード・ポイント範囲内の文字 (. (ピリオド) 文字など) を検索する際には、ストリングをプロセス・コード形式に変換する必要はありません。 個々のバイトを検査して、この文字 (.) を探すだけで十分です。 この例外を利用して、カーネルやユーティリティーでファイル名の構文解析中に特殊文字の . や / を検索できます。プログラムが固有のコード・ポイント範囲内の文字を検索する場合には、バイトに対して操作を行う標準的なストリング関数 (**strchr** サブルーチンなど) を使用する必要があります。 固有のコード・ポイント範囲内の文字のリストについては、 55 ページの『ASCII 文字』を参照してください。

ファイル名の突き合わせ

POSIX.2 は、ファイル名の突き合わせに使用する **fnmatch** サブルーチンを定義しています。 アプリケーションで **fnmatch** サブルーチンを使用して、ディレクトリーを読み取り、個々のエントリーに対してパターンを適用できます。 例えば、 **find** ユーティリティーで **fnmatch** サブルーチンを使用できます。 **pax** ユーティリティーで **fnmatch** サブルーチンを使用してパターン・オペランドを処理できます。同様の方法でストリングを突き合わせなければならないアプリケーションで、 **fnmatch** サブルーチンを使用できます。

基数文字の処理

nl_langinfo(RADIXCHAR**)** で入手される基数文字は、ストリングを指すポインターであることに注意してください。 ロケールでこの文字はマルチバイト文字としてもストリングとしても指定できます。 しかしながら、AIX では、 RADIXCHAR が 1 バイト文字であるという単純な前提事項があります。

プログラミング・モデル

下記のプログラミング・モデルは、既存のプログラムを国際化する場合または新しいプログラムを開発する場合に、加える必要のある変更内容を強調しています。

- 完全なグローバリゼーションを行う。文字に特定の属性があることを想定しないでください。 適切なインターフェースを使用して、動的に属性を決定してください。 固有のコード・ポイント範囲内の ASCII 文字を除いて、コード・セットの属性を想定しないでください。
- プログラムをコード・セットから独立させる。 プログラムが単一バイト、2 バイト、またはどのような種類のマルチバイトのエンコードも想定しないようにしてください。 適切なサブルーチンを使用して、プロセス・コード形式かファイル・コード形式のデータを処理できます。
- ファイル・コード形式に限りカーネルと対話できるようにする。 カーネルはプロセス・コードを処理できません。
- 各国語サポート・サブルーチン・ライブラリーは、プロセス・コードを基にした処理と同様に、ファイル・コードを基にした処理を行える。

  注**:** プロセス・コード形式を基にしたサブルーチンの中には、ファイル・コード形式を基にした対応するサブルーチンがないものもあります。 このように対応していないために、ストリングをプロセス・コードに変換してから、適切なプロセス・コード・サブルーチンを呼び出す必要が生じることがあります。
- ライブラリーの中には、プロセス・コード形式の処理を備えていないものもある。これらのライブラリーを必要とするアプリケーションは、これらのライブラリーから関数を呼び出す際にファイル・コードを使用しなければなりません。
- プログラムはプロセス・コード形式かファイル・コード形式のいずれかの文字を処理できる。 両方のメソッドを使用してコード・セットから独立したプログラムを作成できます。

各国語サポートのサブルーチン

このセクションでは、プログラマーが国際化されたポータブル・プログラムを開発する際にサブルーチンを使用する場合の指針を示します。可搬性を最大限にするには、標準的な Open Group、ISO/ANSI C、および POSIX 関数を使用してください。

注**:** アプリケーションが表示タイプのサービスを行っていない場合は、 **libi18n.a** ライブラリー中のレイアウト・サブルーチンを使用しないでください。 ほとんどのアプリケーションは論理的に配列されたテキストを処理します。

関連概念:

11 ページの『マルチバイト・サブルーチン』

マルチバイト・サブルーチンは、ファイル・コード形式の文字を処理します。普通この種のサブルーチンの名前の先頭は、接頭部 **mb** になります。しかしながら、この接頭部が存在しないマルチバイト・サブルーチンもあります。 例えば、 **strcoll** サブルーチンと **strxfrm** サブルーチンは、マルチバイト形式の文字を処理しますが、 **mb** 接頭部はありません。標準的な C サブルーチン **strcmp**、 **strcpy**、**strncmp**、**strncpy**、**strcat**、および **strncat** は、バイトに対して操作を行い、マルチバイト・データの処理に使用できます。 標準的な C 検索サブルーチン **strchr**、 **strrchr**、**strpbrk**、 **strcspn**、**strrchr**、 **strspn**、**strstr**、および **strtok** は、以下の場合に使用できます。

12 ページの『ワイド文字サブルーチン』
ワイド文字サブルーチンは、プロセス・コード形式の文字を処理します。 普通、ワイド文字サブルーチンの先頭は、接頭部 **wc** になります。しかしながら、この規則には例外があります。 例えば、ワイド文字種別関数は **isw** 接頭部を使用します。 特定のサブルーチンがワイド文字サブルーチンかどうかを判断するには、サブルーチンのプロトタイプで文字が **wchar_t** データ型または **wchar_t** データ・ポインターとして定義されているか、またはサブルーチンが **wchar_t** データ型を戻すかどうか検査してください。 この規則には例外があります。 例えば、ワイド文字種別サブルーチンは **wint_t** データ型値を受け入れます。

196 ページの『各国語サポート・サブルーチンのリスト』
各国語サポート・サブルーチンは、ロケール固有の情報を処理し、ワイド文字およびマルチバイト文字を操作し、正規表現を使用する場合に使用します。

201 ページの『正規表現サブルーチンのリスト』

ロケール・サブルーチン

ロケールに依存した処理 (ユーザー・メッセージを含む) を実行するプログラムは、開始時に **setlocale** サブルーチンを呼び出さなければなりません。 この呼び出しは、主プログラム中の先頭の実行可能ステートメントです。この方法で **setlocale** サブルーチンを呼び出さないプログラムは、 C ロケールか POSIX ロケールを継承します。 この種のプログラムは、 **LC_*** 環境変数や **LANG** 環境変数の設定にかかわらず、C ロケール中の場合と同様に実行します。

その他のサブルーチンは、ロケール・データの形式設定の現行設定を決定するために備えられています。

プロセスのロケールにより、文字照合、文字種別、日付および時刻の形式設定、数字の句読点、通貨の句読点、およびメッセージ出力の処理方法が決まります。 ここでは、各国語サポートを使用してプログラム内の現行ロケールに関する情報を設定してアクセスする方法について説明します。

関連概念:

『ロケールの設定』
すべての国際化されたプログラムは、 **setlocale** サブルーチンを使用して、現行ロケールを設定しなければなりません。プロセスでこのサブルーチンを使用すると、ロケール・データベースにアクセスして現行ロケールの変更や照会を行えます。

ロケールの設定

すべての国際化されたプログラムは、 **setlocale** サブルーチンを使用して、現行ロケールを設定しなければなりません。プロセスでこのサブルーチンを使用すると、ロケール・データベースにアクセスして現行ロケールの変更や照会を行えます。

プロセスを開始すると、現行ロケールが C ロケールか POSIX ロケールに設定されます。 C ロケールか POSIX ロケールに定義されていないロケールに依存しているプログラムが、ロケールごとに固有の情報を使用するには、その前に以下の方法で **setlocale** サブルーチンを呼び出さなければなりません。

```
setlocale(LC_ALL, "");
```

関連概念:

『ロケール・サブルーチン』
ロケールに依存した処理 (ユーザー・メッセージを含む) を実行するプログラムは、開始時に **setlocale** サブルーチンを呼び出さなければなりません。 この呼び出しは、主プログラム中の先頭の実行可能ステートメントです。この方法で **setlocale** サブルーチンを呼び出さないプログラムは、 C ロケールか POSIX ロケールを継承します。 この種のプログラムは、 **LC_*** 環境変数や **LANG** 環境変数の設定にかかわらず、C ロケール中の場合と同様に実行します。

189 ページの『プログラム操作のチェックリスト』

ロケール情報へのアクセス

以下のサブルーチンには、最新の **setlocale** サブルーチン呼び出しによって決定した現行ロケール中に定義されている情報に対するアクセス権があります。

localeconv
現行ロケールの **LC_MONETARY** および **LC_NUMERIC** カテゴリー中に定義されているロケール情報に対するアクセス権を備えています。 **localeconv** サブルーチンは、これらのカテゴリーに関する情報を取り出し、 **locale.h** ファイルに定義されているとおりにタイプ **lconv** の構造体にその情報を挿入し、この構造体を指すポインターを戻します。

nl_langinfo
現行ロケールの **LC_CTYPE**、 **LC_MESSAGES**、 **LC_MONETARY**、 **LC_NUMERIC**、および **LC_TIME** カテゴリー中に定義されている情報を含むヌル終了ストリングを指すポインターを戻します。

rpmatch
現行ロケールの **LC_MESSAGES** カテゴリーに指定されている、肯定応答と否定応答をテストします。 応答は、単純なストリングの場合もあれば、正規表現の場合もあります。 **rpmatch** サブルーチンは業界標準のサブルーチンではないので、ポータブル・アプリケーションでこのサブルーチンが使用可能であると想定しないようにしてください。

localeconv サブルーチンと **nl_langinfo** サブルーチンは、すべての **LC_*** カテゴリーに対するアクセス権を備えている訳ではありません。

カテゴリーの現行ロケール設定を入手するには、 **setlocale**(*Category*, (char*)0) を使用します。戻り値は、 *Category* の現行ロケールを指定したストリングです。以下の例は、 **LC_CTYPE** カテゴリーの現行ロケール設定を判別します。

```
char *ctype_locale; ctype_locale = setlocale(LC_CTYPE, (char*)0);
```

例

このセクションには、サブルーチンの例が含まれています。

- 以下の例は、 **setlocale** サブルーチンを使用して、デフォルトの C ロケールから、環境変数で指定され、ロケール環境変数の階層と整合性のあるロケールに変更します。

```
#include <locale.h>
main()
{

    char *p;

    p = setlocale(LC_ALL, "");

   /*
    **  The program will have the locale as set by the
    **  LC_* and LANG variables.
    */
 }
```

- 以下の例は、 **setlocale** サブルーチンを使用して、 **LC_COLLATE** カテゴリーの現行ロケール設定を入手します。

```
#include <stdio.h>
#include <locale.h>

main()
```

```
{
    char *p;

    /*  set the current locale to what is specified  */
    p = setlocale(LC_ALL, "");
    /*  The current locale settings for all the
    **  categories is pointed to by p
    */

    /*
    **  Find the current setting for the
    **  LC_COLLATE category
    */
    p = setlocale(LC_COLLATE, NULL);
    /*
    **  p points to a string containing the current locale
    **  setting for the LC_COLLATE category.
    */

 }
```

- 以下の例は、 **setlocale** サブルーチンを使用して、現行ロケール設定を入手し、後で使用するために保管します。 プログラムでこのアクションを使用すると、ロケールを一次的に新しいロケールに変更できます。 処理の完了後に、ロケールは元の状態に戻ります。

```
#include <stdio.h>
#include <locale.h>
#include <string.h>

#define NEW_LOCALE "MY_LOCALE"

main()
{
    char *p, *save_locale;

    p = setlocale(LC_ALL, "");
    /*
    **  Initiate locale. p points to the current locale
    **  setting for all the categories
    */

    save_locale = (char *)malloc(strlen(p) +1);
    strcpy(save_locale, p);
        /*  Save the current locale setting */
    p = setlocale(LC_ALL, NEW_LOCALE);
        /* Change to new locale */

    /*
    **  Do processing  ...
    */

     /*  Change back to old locale  */
    p = setlocale(LC_ALL, save_locale);  /* Restore old locale */

     free(save_locale);    /* Free the memory */
}
```

- 以下の例は、 **setlocale** サブルーチンを使用して、 **LC_MESSAGES** カテゴリーを環境変数によって決められているロケールに設定します。 その他のカテゴリーはすべて C ロケールに設定されたままになります。

```
#include <locale.h>

main()
{
    char *p;
```

```
    /*
    **  The program starts in the C locale for all categories.
    */

    p = setlocale(LC_MESSAGES, "");

     /*
    **  At this time the LC_COLLATE, LC_CTYPE, LC_NUMERIC,
    **  LC_MONETARY, LC_TIME will be in the C locale.
    **  LC_MESSAGES will be set to the current locale setting
    **  as determined by the environment variables.
    */
}
```

- 以下の例は、 **localeconv** サブルーチンを使用して、現行ロケールの小数点設定を入手します。

```
#include <locale.h>

main()
{
        struct lconv *ptr;
        char *decimal;

        (void)setlocale(LC_ALL, "");
        ptr = localeconv();
        /*
        ** Access the data obtained. For example,
        ** obtain the current decimal point setting.
        */
        decimal = ptr->decimal_point;
}
```

- 以下の例は、 **nl_langinfo** サブルーチンを使用して、現行ロケールの日付および時刻形式を入手します。

```
#include <langinfo.h>
#include <locale.h>
main()
{
    char *ptr;
    (void)setlocale(LC_ALL, "");
    ptr = nl_langinfo(D_T_FMT);
}
```

- 以下の例は、 **nl_langinfo** サブルーチンを使用して、現行ロケールの基数文字を入手します。

```
#include <langinfo.h>
#include <locale.h>

main()
{
    char *ptr;
    (void)setlocale(LC_ALL, "");  /* Set the program's locale  */
    ptr = nl_langinfo(RADIXCHAR); /* Obtain the radix character*/
}
```

- 以下の例は、 **nl_langinfo** サブルーチンを使用して、現行ロケールの通貨記号の設定を入手します。

```
#include <langinfo.h>
#include <locale.h>

main()
{
    char *ptr;
    (void)setlocale(LC_ALL, "");  /* Set the program's locale  */
    ptr = nl_langinfo(CRNCYSTR);  /* Obtain the currency string*/
    /* The currency string will be "-$" in the U. S. locale. */
}
```

- 以下の例は、 rpmatch サブルーチンを使用して、現行ロケールの肯定応答と否定応答のストリングの設定を入手します。

  ロケール・データベース中で指定されている肯定応答と否定応答は、単純なストリングではなくなり、正規表現にすることができます。 例えば、yesexpr を以下のような正規表現にすることができます。この式には、大文字と小文字の y の後にゼロ個以上の英字か、文字 O の後に K を使用できます。したがって、yesexpr を以下のような正規表現にすることができます。

  ```
  ([yY][:alpha:]*|OK)
  ```

  この標準には、この情報を取り出して比較するサブルーチンは含まれていません。 AIX 固有の **rpmatch(const char *response)** サブルーチンを使用できます。

  ```
  #include <stdio.h>
  #include <langinfo.h>
  #include <locale.h>
  #include <regex.h>

  int rpmatch(const char *);
      /*
      **  Returns 1 if yes response, 0 if no  response,
      ** -1 otherwise
      */

  main()
  {
      int  ret;
      char *resp;

  (void)setlocale(LC_ALL, "");

      do {
          /*
          ** Obtain the response to the query for yes/no strings.
          ** The string pointer resp points to this response.
          ** Check if the string is yes.
          */
      ret = rpmatch(resp);

      if(ret == 1){
          /* Response was yes. */
          /* Process accordingly. */
          }else if(ret == 0){
          /* Response was negative. */
          /* Process negative response. */
          }else if(ret<0){
          /* No match with yes/no occurred. */
          continue;
          }
      }while(ret <0);
  }
  ```

- 以下の例は、 **rpmatch** サブルーチンをインプリメントするメソッドを備えます。ほとんどのアプリケーションでは、 **libc** 中に **rpmatch** サブルーチンを使用する必要があることに注意してください。 以下の **rpmatch** サブルーチンをインプリメントしたものは、図示する目的のためだけに記載されています。

  nl_langinfo(YESEXPR) は肯定応答の正規表現を入手するのに使用し、 **nl_langinfo(NOEXPR)** は否定応答の正規表現を入手するのに使用することに注意してください。

  ```
  #include <langinfo.h>
  #include <regex.h>
  /*
  ** rpmatch() performs comparison of a string to a regular expression
  ```

```
** using the POSIX.2 defined regular expression compile and match
** functions. The first argument is the response from the user and the
** second string is the current locale setting of the regular expression.
*/
int rpmatch( const char *string)



{
     int status;
     int retval;
     regex_t re;
     char *pattern;

     pattern = nl_langinfo(YESEXPR);
     /* Compile the regular expression pointed to by pattern. */
     if( ( status = regcomp( &re, pattern, REG_EXTENDED | REG_NOSUB )) != 0 ){
          retval = -2; /*-2 indicates yes expr compile error */
          return(retval);
     }
     /* Match the string with the compiled regular expression. */
     status = regexec( &re, string, (size_t)0, (regmatch_t *)NULL, 0);
     if(status == 0){
          retval = 1;   /* Yes match found */
     }else{    /* Check for negative response */
          pattern = nl_langinfo(NOEXPR);
          if( ( status = regcomp( &re, pattern,
               REG_EXTENDED | REG_NOSUB )) != 0 ){
                    retval = -3;/*-3 indicates no compile error */
                    return(retval);
               }
          status = regexec( &re, string, (size_t)0,
                         (regmatch_t *)NULL, 0);
          if(status == 0)
               retval = 0;/* Negative response match found */
     }else
               retval = -1; /* The string did not match yes or no
                               response */
     regfree(&re);
     return(retval);
}
```

時刻形式設定サブルーチン

時刻をワイド文字コードのストリングに形式設定する必要のあるプログラムでは、 **wcsftime** サブルーチンを使用できます。 マルチバイト・ストリングを内部時刻形式に変換する必要のあるプログラムでは、 **strptime** サブルーチンを使用できます。

C プログラミング言語標準で定義されている **strftime** サブルーチンに加えて、 X/Open Portability Guide Issue 4 では、以下の時刻形式設定サブルーチンが定義されています。

wcsftime
時刻をワイド文字コードのストリングに形式設定する。

strptime
マルチバイト・ストリングを内部時刻形式に変換する。

例

次の例では、 **wcsftime** サブルーチンを使用して、時刻をワイド文字ストリングに形式設定します。

多くの場合、「これは何ですか?」という質問に答えて、定義で始まります。

```
#include <stdio.h>
#include <langinfo.h>
#include <locale.h>
#include <time.h>

main()
{
        wchar_t timebuf[BUFSIZE];
        time_t clock = time( (time_t*) NULL);
        struct tim *tmptr = localetime(&clock);

        (void)setlocale(LC_ALL, "");

        wcsftime(
                timebuf,       /* Time string output buffer */
                BUFSIZ,        /*Maximum size of output string */
                nl_langinfo(D_T_FMT),      /* Date/time format */
                tmptr          /* Pointer to tm structure */
        );

        printf("%S¥n", timebuf);
}
```

次の例では、 strptime サブルーチンを使用して、形式設定済みのタイム・ストリングを内部形式に変換します。

```
#include <langinfo.h>
#include <locale.h>
#include <time.h>

main(int argc, char **argv)
{
        struct tm tm;

        (void)setlocale(LC_ALL, "");

        if (argc != 2) {
                ...                /* Error handling */
        }
        if (strptime(
                argv[1],           /* Formatted time string */
                nl_langinfo(D_T_FMT),      /* Date/time format */
                &tm                /* Address of tm structure */
        ) == NULL) {
                ...                /* Error handling */
        }
        else {
                ...               /* Other Processing */
        }
}
```

通貨形式設定サブルーチン

通貨の金額を指定したり、それにアクセスしたりする必要のあるプログラムは、**strfmon** サブルーチンを呼び出せます。

C プログラミング言語標準を POSIX と共に使用すると、通貨情報の指定やアクセスの手段を提供しますが、これらの標準では通貨の金額を形式設定するサブルーチンは定義されていません。XPG4 **strfmon** サブルーチンには、通貨の金額を形式設定する機能は備えられていません。 形式設定済みの通貨ストリングを数字の金額に変換して演算できるようにするサブルーチンは定義されていません。 通貨の金額に関する演算を行う必要のあるアプリケーションは、ロケールに依存する通貨ストリングを数値に変換処理した後でこの種の演算を行うことができます。国/地域別の通貨の形式設定に関する情報は、 **LC_MONETARY** カ

テゴリーで指定されています。 アプリケーションで、**localeconv** サブルーチンを呼び出して、通貨の形式や通貨記号に関連する情報を入手することができます。

注: POSIX ロケールには、@euro や @preeuro の修飾子は不要です。

例

以下の例は、 **strfmon** サブルーチンを使用して、形式の仕様と入力値を受け入れます。 入力値は、入力形式の仕様に従って形式設定されます。

```
#include <monetary.h>
#include <locale.h>
#include <stdio.h>

main(int argc, char **argv)
{
        char bfr[256], format[256];
        int match; ssize_t size;
        float value;

        (void) setlocale(LC_ALL, "");

        if (argc != 3){
                ...        /* Error handling */
        }
        match = sscanf(argv[1], "%f", &value);
        if (!match) {
                ...        /* Error handling */
        }
        match = sscanf(argv[2], "%s", format);
        if (!match) {
                ...        /*Error handling */
        }
        size = strfmon(bfr, 256, format, value);
        if (size == -1) {
                ...        /* Error handling */
        }
        printf ("Formatted monetary value is: %s¥n", bfr);
}
```

以下の表は、米国英語ロケールでの 12345.67 と -12345.67 の変換仕様と出力として有効なその他の例を示しています。

変換の仕様	出力	出力
%n	$12,345.67 -$12,345.67$12	デフォルトの形式設定
%15n	$12,345.67 -$12,345.67	15 文字のフィールド中で右寄せされます。
%#6n	$ 12,345.67 -$ 12,345.67	最大 999,999 までの値の列を位置合わせします。
%=*#8n	$****12,345.67 -$****12,345.67	充てん文字を指定します。
%=#8n	$000012,345.67 -$000012,345.67	充てん文字にグループ分けを使用しません。
%^#6n	$ 12345.67 -$ 12345.67	3 桁ごとの区切りを使用不可にします。
%^#6.0n	12346 -$ 12346	自然数の単位に四捨五入します。
%^#6.3n	$ 12345.670 -$ 12345.670	精度を増します。
%(#6n	$ 12,345.67 ($ 12,345.67)	代替の肯定または否定のスタイルを使用します。
%!(#6n	12,345.67 (12,345.67)	通貨記号を使用不可にします。

以下の例は、通貨の値を数値に変換します。 通貨ストリングが input によって指され、数字形式への変換結果は output によって指されたストリングに格納されます。 input と output が初期設定されていることが前提になっています。

```
char *input;   /* the input multibyte string containing the monetary string */
char *output;  /* the numeric string obtained from the input string */
wchar_t src_string[SIZE], dest_string[SIZE];
wchar_t *monetary, *numeric;
wchar_t mon_decimal_point, radixchar;
wchar_t wc;
localeconv *lc;

/* Initialize input and output to point to valid buffers as appropriate. */
/* Convert the input string to process code form*/
retval = mbstowcs(src_string, input, SIZE);
/* Handle error returns */

monetary = src_string;
numeric = dest_string;
lc = localeconv();
        /* obtain the LC_MONETARY and LC_NUMERIC info */

/* Convert the monetary decimal point to wide char form */
retval = mbtowc( &mon_decimal_point, lc->mon_decimal_point,
                MB_CUR_MAX);
/* Handle any error case */

/* Convert the numeric decimal point to wide char form */
retval = mbtowc( &radixchar, lc->decimal_point, MB_CUR_MAX);
/* Handle error case */
/* Assuming the string is converted first into wide character
** code form via mbstowcs, monetary points to this string.
*/
/* Pick up the numeric information from the wide character
** string and copy it into a temp buffer.
*/
        while(wc = *monetary++){
                if(iswdigit(wc))
                        *numeric++ = wc;
                else if( wc == mon_decimal_point)
                        *numeric++=radixchar;
        }
        *numeric = 0;
/* dest_string has the numeric value of the monetary quantity. */
/* Convert the numeric quantity into multibyte form */
retval = wcstombs( output, dest_string, SIZE);
/* Handle any error returns */
/* Output contains a numeric value suitable for atof conversion. */
```

マルチバイト・サブルーチンとワイド文字サブルーチン

データの外部表現のことを、文字のファイル・コード 表現といいます。 ファイル・コード・データをファイル中に作成したりコンピューターとその入出力デバイスとの間で転送したりする際に、 1 文字を単一バイトか複数のバイトで表すことができます。 この種の文字のストリングを処理する場合は、これらのコードを一様な長さの表現に変換した方が効率的です。 変換後の形式は、文字の内部処理に使用できます。 データの内部表現のことを、文字のプロセス・コード またはワイド文字コード 表現といいます。

各国語サポートでは、マルチバイト・サブルーチンとワイド文字サブルーチンを併用して、プログラムのグローバリゼーションを行います。マルチバイト ・サブルーチンは、マルチバイト文字セットを使用します。 ワイド文字 サブルーチンは、ワイド文字セットを使用します。 マルチバイト・サブルーチンには **mb** 接頭部があります。ワイド文字サブルーチンには wc 接頭部があります。 対応するストリング処理サ

ブルーチンは、それぞれ **mbs** 接頭部と **wcs** 接頭部で示されます。 マルチバイト・サブルーチンやワイド文字サブルーチンを使用する時点を決める際には、必ずその前に注意深く分析するようにしてください。

マルチバイト・コードと文字コードの変換サブルーチン

各国語サポートの国際化環境では、マルチバイト・サブルーチンとワイド文字サブルーチンが併用されます。ワイド文字サブルーチンやマルチバイト・サブルーチンを使用する時点を決める際には、必ずその前に注意深く分析するようにしてください。

プログラムで初めてマルチバイト・サブルーチンを使用する際に、特定のワイド文字サブルーチンを使用できるようにするには、その前にマルチバイト文字コードをワイド文字コードに変換する必要が生じることがあります。 プログラムでワイド文字サブルーチンを使用する際に、サブルーチンを呼び出すには、データをマルチバイト形式に変換する必要が生じることがあります。 どちらの方式も、使用中のプログラムや、必要な処理を実行する標準サブルーチンが使用できるかどうかに応じて、欠点があります。 例えば、ワイド文字の表示列幅サブルーチンには、対応する標準的なマルチバイト・サブルーチンはありません。

プログラムでマルチバイト形式の文字を処理できる場合は、文字をワイド文字形式に変換する代わりにこの方式を使用する必要があります。

重要**:** マルチバイト・コードとワイド文字コードの間の変換は、現行ロケールの設定に応じて異なります。 使用しようとしている個々のロケールが一貫性のある方法でワイド文字コードを処理することが分かっていない場合は、 2 つのプロセス間でワイド文字コードを交換しないでください。 IBM-eucTW コード・セットに基づいたロケールを除いて、 AIX のロケールはワイド文字コードとしてユニコード文字値を使用します。

マルチバイト・コードからワイド文字コードへの変換サブルーチン**:**

以下のサブルーチンは、マルチバイト・コードからワイド文字コードに変換する際に使用します。

mblen
: マルチバイト文字の長さを判別する。 p++ を使用して、マルチバイト・ストリング中のポインターを増分しないでください。 **mblen** サブルーチンを使用して、文字を構成するバイト数を決定してください。

mbstowcs
: マルチバイト・ストリングをワイド文字ストリングに変換する。

mbtowc
: マルチバイト文字をワイド文字に変換する。

ワイド文字コードからマルチバイト・コードへの変換サブルーチン**:**

以下のサブルーチンは、ワイド文字コードからマルチバイト文字コードに変換する際に使用します。

wcslen
: ワイド文字ストリング中のワイド文字の数を決定する。

wcstombs
: ワイド文字ストリングをマルチバイト文字のストリングに変換する。

wctomb
: ワイド文字をマルチバイト文字に変換する。

例:

次の例では、 **mbtowc** サブルーチンを使用して、マルチバイト文字コードの文字をワイド文字コードに変換します。

```
main()
{
    char     *s;
    wchar_t  wc;
    int      n;

    (void)setlocale(LC_ALL,"");

    /*
    **  s points to the character string that needs to be
    **  converted to a wide character to be stored in wc.
    */
    n = mbtowc(&wc, s, MB_CUR_MAX);

    if (n == -1){
        /*  Error handle  */
    }
    if (n == 0){
        /*  case of name pointing to null  */
    }

    /*
    **  wc contains the process code for the multibyte character
    **  pointed to by s.
    */
}
```

次の例では、 **wctomb** サブルーチンを使用して、ワイド文字ストリングの文字をマルチバイト文字コードに変換します。

```
main()
{
    char     *s;
    wchar_t  wc;
    int      n;

    (void)setlocale(LC_ALL,"");

    /*
    **  s points to the character string that needs to be
    **  converted to a wide character to be stored in wc.
    */
    n = mbtowc(&wc, s, MB_CUR_MAX);

    if (n == -1){
        /*  Error handle  */
    }
    if (n == 0){
        /*  case of name pointing to null  */
    }

    /*
    **  wc contains the process code for the multibyte character
    **  pointed to by s.
    */
}
```

次の例では、 mblen サブルーチンを使用して、マルチバイト文字コードの文字のバイト長を検索します。

```
#include <stdlib.h>
#include <locale.h>

main
{
    char *name = "h";
    int  n;

    (void)setlocale(LC_ALL,"");

    n = mblen(name, MB_CUR_MAX);
    /*
    **  The count returned in n is the multibyte length.
    **  It is always less than or equal to the value of
    **  MB_CUR_MAX in stdlib.h
    */
    if(n == -1){
        /* Error Handling */
    }
}
```

次の例では、マルチバイト・ストリング中の直前の文字位置を入手します。 直前の文字位置を判別する必要がある場合は、現行文字位置 (ランダムなバイト位置ではない) を始点として、バッファーを最初からステップスルーしてください。 現行文字位置に達するまで **mblen** サブルーチンを使用し、直前の文字位置を保管して、必要な文字位置を入手してください。

```
char buf[];     /* contains the multibyte string */
char *cur,      /* points to the current character position */
char *prev,     /* points to previous multibyte character */
char *p;        /* moving pointer */

/* initialize the buffer and pointers as needed */
/* loop through the buffer until the moving pointer reaches
** the current character position in the buffer, always
** saving the last character position in prev pointer */
p = prev = buf;

/* cur points to a valid character somewhere in buf */
while(p< cur){
        prev = p;
        if( (i=mblen(p, mbcurmax))<=0){
                /* invalid multibyte character or null */
                /* You can have a different error handling
                ** strategy */
                p++;    /* skip it */
        }else {
                p += i;
        }
}
/* prev will point to the previous character position */

/* Note that if( prev == cur), then it means that there was
** no previous character. Also, if all bytes up to the
** current character are invalid, it will treat them as
** all valid single-byte characters and this may not be what
** you want. One may change this to handle another method of
** error recovery. */
```

次の例では、 **mbstowcs** サブルーチンを使用して、マルチバイト・ストリングをワイド文字ストリングに変換します。

```
#include <stdlib.h>
#include <locale.h>

main()
{
```

```
     char    *s;
     wchar_t *pwcs;
     size_t  retval, n;

     (void)setlocale(LC_ALL, "");

     n = strlen(s) + 1;          /*string length + terminating null */

     /*  Allocate required wchar array    */
     pwcs = (wchar_t *)malloc(n * sizeof(wchar_t) );
     retval = mbstowcs(pwcs, s, n);
     if(retval == -1){

     /*  Error handle  */
          }
          /*
          ** pwcs contains the wide character string.
          */
}
```

次の例では、 **mbstowcs** サブルーチンを使用して大規模なデータのブロックをワイド文字の形式に変換する際の問題を図示しています。 無効なマルチバイトが検出されると、 **mbstowcs** サブルーチンは -1 の値を戻しますが、エラーが発生した場所は指定されません。したがって、 **mbtowc** サブルーチンを繰り返し使用して、一度に 1 文字ずつワイド文字コードに変換しなければなりません。

注: この方法で処理すると、プログラムのパフォーマンスが非常に遅くなります。

単一バイトのコード・セットの変換中に、部分マルチバイトが生じる可能性はありません。 しかしながら、マルチバイト・コード・セットの変換中には、部分マルチバイトが保管バッファーにコピーされます。次回 **read** サブルーチンを呼び出す際に、部分マルチバイトは残りのバイト・シーケンスの接頭部になります。

注: ヌル終了のワイド文字ストリングが取得されます。 無効なバイト・シーケンスのインスタンスが認識された場合には、オプションのエラー処理を行えます。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main(int argc, char *argv[])
{

    char     *curp, *cure;
    int      bytesread, bytestoconvert, leftover;
    int      invalid_multibyte, mbcnt, wcnt;
    wchar_t  *pwcs;
    wchar_t  wbuf[BUFSIZ+1];
    char     buf[BUFSIZ+1];
    char     savebuf[MB_LEN_MAX];
    size_t   mb_cur_max;
    int      fd;
        /*
        **  MB_LEN_MAX specifies the system wide constant for
        **  the maximum number of bytes in a multibyte character.
        */

    (void)setlocale(LC_ALL, "");
    mb_cur_max = MB_CUR_MAX;

    fd = open(argv[1], 0);
    if(fd < 0){
       /*  error handle  */
    }
```

```
    leftover = 0;
    if(mb_cur_max==1){    /*  Single byte code sets case  */
        for(;;){
            bytesread = read(fd, buf, BUSIZ);
            if(bytesread <= 0)
                break;
            mbstowcs(wbuf, buf, bytesread+1);
            /*  Process using the wide character buffer   */
        }
                        /*  File processed ...   */
        exit(0);        /*  End of program       */

    }else{              /*  Multibyte code sets  */
        leftover = 0;

        for(;;) {
            if(leftover)
                strncpy(buf, savebuf ,leftover);
            bytesread=read(fd,buf+leftover, BUFSIZ-leftover);
            if(bytesread <= 0)
                break;

            buf[leftover+bytesread] = '¥0';
                     /* Null terminate string */
            invalid_multibyte = 0;
            bytestoconvert = leftover+bytesread;
            cure= buf+bytestoconvert;
            leftover=0;
            pwcs = wbuf;
                /* Stop processing when invalid mbyte found. */
            curp= buf;

            for(;curp<cure;){
                mbcnt = mbtowc(pwcs,curp, mb_cur_max);
                if(mbcnt>0){
                    curp += mbcnt;
                    pwcs++;
                    continue;

                }else{
                    /* More data needed on next read*/
                    if ( cure-curp<mb_cur_max){
                        leftover=cure-curp;
                        strncpy(savebuf,curp,leftover);
                        /* Null terminate before partial mbyte */
                        *curp=0;
                        break;

                    }else{
                             /*Invalid multibyte found */
                        invalid_multibyte =1;
                        break;
                    }
                }
            }
            if(invalid_multibyte){          /*error handle */
            }
            /* Process the wide char buffer */
        }
    }
}
```

次の例では、 **wcstombs** サブルーチンと **wcslen** サブルーチンを使用して、ワイド文字ストリングをマルチバイト形式に変換します。

```
#include <stdlib.h>
#include <locale.h>

main()
{
    wchar_t *pwcs; /* Source wide character string */
    char *s;       /* Destination multibyte character string */
    size_t n;
    size_t retval;

    (void)setlocale(LC_ALL, "");
    /*
    ** Calculate the maximum number of bytes needed to
    ** store the wide character buffer in multibyte form in the
    ** current code page and malloc() the appropriate storage,
    ** including the terminating null.
    */
    s = (char *) malloc( wcslen(pwcs) * MB_CUR_MAX + 1 );
    retval= wcstombs( s, pwcs, n);
    if( retval == -1) {
        /* Error handle */
        /* s points to the multibyte character string. */
}
```

ワイド文字種別サブルーチン

大多数のワイド文字種別サブルーチンは従来の文字種別サブルーチンに似ていますが、例外としてワイド文字種別サブルーチンは **wint_t** データ型引数として渡される **wchar_t** データ型引数に対して操作を行います。

関連概念:

189 ページの『プログラム操作のチェックリスト』

汎用ワイド文字種別サブルーチン:

グローバリゼーションの環境では、新しい文字クラス属性を作成する機能が必要です。 例えば、日本語の文字には複数の属性が定義されますが、これらの属性は英語の言語に適用できません。 サポートされる言語が増えるにつれて数が変わる文字属性を、アプリケーションで処理できるようにするフレームワークが必要になります。

wctype サブルーチンと **iswctype** サブルーチンを使用すると、一般的な方法で文字クラスを処理できます。 これらのサブルーチンを使用して、ユーザー定義の文字クラスと言語ごとに固有の文字クラスの両方とも処理できます。

wctype サブルーチンや **iswctype** サブルーチンで使用する新しい文字種別を作成するには、 **LC_CTYPE** カテゴリー中に新しい文字クラスを作成し、 **localedef** コマンドを使用してロケールを生成します。 ユーザー・アプリケーションは、 **setlocale** サブルーチンを使用してこのロケール・データを入手します。 それからプログラムは、 **wctype** サブルーチンを使用して **wctype_t** 属性ハンドルを取得し、新しい種別にアクセスします。 それから属性ハンドルとテストされるワイド文字の文字コードの両方を、 **iswctype** サブルーチンに渡します。

以下のサブルーチンをワイド文字種別に使用します。

wctype
: 文字属性種別のハンドルを入手する。

iswctype
: 文字属性をテストする。

標準的なワイド文字種別サブルーチン:

isw* サブルーチンは、標準的なワイド文字種別のさまざまな局面を決定します。 **isw*** サブルーチンは、単一バイトのコード・セットも処理します。 **isw*** サブルーチンは、 **wctype** サブルーチンや **iswctype** サブルーチンに優先して使用してください。 **wctype** サブルーチンと **iswctype** サブルーチンは、外字クラス属性 (日本語の属性など) 専用として使用してください。

ワイド文字関数を使用して、複数のデータ・ブロック中の大/小文字を変換する際には、アプリケーションで文字をマルチバイトからワイド文字コード形式に変換しなければなりません。 この作業により単一バイト・コード・セットのロケールのパフォーマンスに影響することがあるので、アプリケーション中に 2 つの変換パスを設けることを考慮してください。 従来のパスは単一バイト・コード・セットのロケール用で、 **isupper**、**islower**、**toupper**、および **tolower** サブルーチンを使用して大/小文字を変換します。 代替パスはマルチバイト・コード・セットのロケール用で、 **iswupper**、**iswlower**、**towupper**、および **towlower** サブルーチンを使用して、マルチバイト文字をワイド文字コード形式に変換し、大/小文字を変換します。マルチバイト文字をワイド文字コード形式に変換する際には、連続するブロック間でマルチバイト文字が分割されているという特殊な事例をアプリケーションが処理できるようにする必要があります。

標準的なワイド文字種別サブルーチンを以下にリストします。

iswalnum
英数字の文字種別をテストする。

iswcntrl
英字の文字種別をテストする。

iswalpha
制御文字の種別をテストする。

iswdigit
数字の文字種別をテストする。

iswgraph
図形文字の種別をテストする。

iswlower
英小文字の文字種別をテストする。

iswprint
印刷可能文字の種別をテストする。

iswpunct
句読文字の種別をテストする。

iswspace
スペース文字の種別をテストする。

iswupper
英大文字の種別をテストする。

iswxdigit
16 進数字の文字種別をテストする。

ワイド文字の大/小文字変換サブルーチン:

以下のサブルーチンは、ワイド文字の大/小文字を変換します。ワイド文字大/小文字変換サブルーチンのアクションは、現行ロケールの **LC_CTYPE** カテゴリー中の定義によって制御されます。

towlower
大文字のワイド文字を小文字のワイド文字に変換する。

towupper
小文字ワイド文字を大文字ワイド文字に変換する。

例:

次の例では、**wctype** サブルーチンを使用して、 **NEW_CLASS** 文字の種別をテストします。

```
#include <ctype.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wint_t    wc;
    int       retval;
    wctype_t  chandle;

    (void)setlocale(LC_ALL,"");
    /*
    ** Obtain the character property handle for the NEW_CLASS
    ** property.
    */
    chandle = wctype("NEW_CLASS") ;
    if(chandle == (wctype_t)0){
        /* Invalid property. Error handle. */
    }
    /* Let wc be the wide character code for a character */
    /* Test if wc has the property of NEW_CLASS */
    retval = iswctype( wc, chandle );
    if( retval > 0 ) {
        /*
        ** wc has the property NEW_CLASS.
        */
    }else if(retval == 0) {
        /*
        ** The character represented by wc does not have the
        ** property NEW_CLASS.
        */
    }
}
```

ワイド文字の表示列幅サブルーチン

文字を表示したり印刷したりする際に、 1 文字に使用される列の数は変わることがあります。 例えば、漢字文字 (日本語) には複数の列位置が使用されます。 文字当たりに必要な表示列の数は、各国語サポートのロケール・データベースで決められます。 **LC_CTYPE** カテゴリーは、文字の表示に必要な列の数を定義します。

標準的なマルチバイト表示列幅サブルーチンはありません。 可搬性を保つために、マルチバイト・コードをワイド文字コードに変換し、必須のワイド文字表示幅サブルーチンを使用してください。 しかしながら、 **__max_disp_width** マクロ (**stdlib.h** ファイル中に定義されている) が 1 に設定されており、単一バイト・コード・セットが使用中である場合は、コード・セット中のすべての文字 (タブを除く) の表示列幅は同じで、1 になります。この場合、 **strlen (***string***)** サブルーチンを使用すると、以下の例のように指定されたストリングの表示列幅を求められます。

```
#include <stdlib.h>
        int display_column_width;  /* number  of  display  columns  */
        char  *s;                  /*  character  string           */
        ....
```

```
        if((MB_CUR_MAX  ==  1)  &&  (__max_disp_width  ==  1)){
                display_column_width  =  strlen(s);
                                /*  s  is  a  string  pointer    */
        }
```

以下のサブルーチンは、ワイド文字ストリングの表示幅を検索します。

wcswidth
ワイド文字ストリングの表示幅を判別する。

wcwidth
ワイド文字の表示幅を判別する。

例:

次の例では、 **wcwidth** サブルーチンを使用して、ワイド文字の表示列幅を検索します。

```
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

main()
{
    wint_t  wc;
    int     retval;

    (void)setlocale(LC_ALL,  "");

    /*
    **    Let wc be the wide character whose display width is
    **    to be found.
    */
    retval  =  wcwidth(wc);
    if(retval  ==  -1){
        /*
        **  Error handling. Invalid or nonprintable
        **  wide character in wc.
        */
    }
}
```

次の例では、 **wcswidth** サブルーチンを使用して、ワイド文字ストリングの表示列幅を検索します。

```
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

main()
{
    wchar_t  *pwcs;
    int      retval;
    size_t   n;

    (void)setlocale(LC_ALL,  "");
    /*
    **    Let pwcs point to a wide character null
    **    terminated string.
    **    Let n be the number of wide characters
    **    whose display column width is to be determined.
    */
    retval  =  wcswidth(pwcs,  n);
    if(retval  ==  -1){
        /*
        **  Error handling. Invalid wide or nonprintable
        **  character  ode encountered in the wide
```

```
        **  character string pwcs.
        */
    }
}
```

マルチバイト文字とワイド文字のストリング照合サブルーチン

以下の方法でストリングを比較できます。

- 文字の序数 (2 進数) 値を使用する。
- **LC_COLLATE** カテゴリーで決定される、ロケールごとの文字に関連した重みを使用する。

各国語サポートでは 2 つ目の方法が使用されます。

照合は、ロケールごとに固有の文字属性です。 重みは、個々の文字に割り当てられ、ソート用の相対順序を示します。 1 文字に複数の重みを割り当てることができます。 重みには 1 次、2 次、3 次と優先順位付けしていきます。 個々の文字に割り当てられる重みの最大数は、システム定義です。

プロセスは起動時に C ロケールか POSIX ロケールを継承します。 **setlocale (LC_ALL, " ")** サブルーチンが呼び出されると、プロセスは **LC_*** 環境変数と **LANG** 環境変数に基づいてロケールを入手します。 以下のサブルーチンは、 **LC_COLLATE** カテゴリーによって制御され、特定のロケール中で 2 つのストリングをソートする方法を決定します。

注**:** 照合値の入手に関係する処理のために、照合に基づくストリングの比較には長時間を要します。 この種の比較は、必要な場合のみ実行してください。 2 つのワイド文字ストリングが等しいかどうかを判別する必要がある場合は、 **wcscoll** サブルーチンや **wcsxfrm** サブルーチンを使用しないでください。代わりに **wcscmp** サブルーチンを使用してください。

以下のサブルーチンは、マルチバイト文字のストリングを比較します。

strcoll
: マルチバイト文字のストリングの照合重みを比較する。

strxfrm
: マルチバイト文字のストリングを、文字照合重みを表す値に変換する。

以下のサブルーチンは、ワイド文字ストリングを比較します。

wcscoll
: ワイド文字ストリングの照合の重みを比較する。

wcsxfrm
: ワイド文字ストリングを文字の照合の重みを表す値に変換する。

例**:**

次の例では、 **wcscoll** サブルーチンを使用して、 2 つのワイド文字ストリングを照合重みに基づいて比較します。

```
#include  <stdio.h>
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

extern  int  errno;

main()
{
    wchar_t  *pwcs1, *pwcs2;
```

```
    size_t   n;

    (void)setlocale(LC_ALL,  "");

    /*    set it to zero for checking errors on wcscoll    */
    errno  =  0;
    /*
    **    Let pwcs1 and pwcs2 be two wide character strings to
    **    compare.
    */
    n  =  wcscoll(pwcs1, pwcs2);
        /*
        **    If errno is set then it indicates some
        **    collation error.
        */
    if(errno  !=  0){
        /*  error has occurred... handle error ...*/
    }
}
```

次の例では、 **wcsxfrm** サブルーチンを使用して、 2 つのワイド文字ストリングを照合重みに基づいて比較します。

注**: wcsxfrm** サブルーチンを使用する場合、以下のいずれかの方法で、変換したストリングのサイズ *n* (*n* は数値) を判別できます。

- ワイド文字ストリング中の文字ごとに、有効な照合値のバイト数は、 **COLL_WEIGHTS_MAX * sizeof(wchar_t)** 値以下である。 この値にワイド文字コードの数を乗算すると、必要なバッファー長が求められます。 このバッファー長に、ワイド文字の終了ヌルの分の 1 を加算します。 このストラテジーを行うと、パフォーマンスがスローダウンします。
- 必要なバイト長を見積もる。 入手済みの値が小さい場合は、大きくします。 この方法はすべてのストリングに適している訳ではありませんが、パフォーマンスは最大になります。
- **wcsxfrm** サブルーチンを 2 回呼び出す。 1 回目は *n* の値を検索し、 2 回目はこの *n* 値を使用してストリングを変換します。このストラテジーは、 **wcsxfrm** サブルーチンを 2 回呼び出すので、パフォーマンスはスローダウンします。 しかしながら、変換したストリングの保管に必要なバッファー・サイズの正確な値を求められます。

プログラム中で使用するストリングの特性と、プログラムのパフォーマンス上の目標に応じて、方式を選択します。

```
#include  <stdio.h>
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

main()
{
    wchar_t  *pwcs1, *pwcs2, *pwcs3, *pwcs4;
    size_t   n, retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  Let the string pointed to by pwcs1 and pwcs3 be the
    **  wide character arrays to store the transformed wide
    **  character strings. Let the strings pointed to by pwcs2
    **  and pwcs4 be the wide character strings to compare based
    **  on the collation values of the wide characters in these
    **  strings.
    **  Let n be large enough (say,BUFSIZ) to transform the two
    **  wide character strings specified by pwcs2 and pwcs4.
    **
```

```
    **  Note:
    **  In practice, it is best to call wcsxfrm if the wide
    **  character string is to be compared several times to
    **  different wide character strings.
    */

    do {
        retval = wcsxfrm(pwcs1, pwcs2, n);
        if(retval == (size_t)-1){
            /*  error has occurred.  */
            /*  Process the error if needed  */
            break;
        }

        if(retval >= n ){
        /*
        ** Increase the value of n and use a bigger buffer pwcs1.
        */
        }
    }while (retval >= n);

    do {
        retval = wcsxfrm(pwcs3, pwcs4, n);
        if (retval == (size_t)-1){
            /*  error has occurred.  */
            /*  Process the error if needed  */
            break;

        if(retval >= n){
        /*Increase the value of n and use a bigger buffer pwcs3.*/
        }
    }while (retval >= n);
    retval = wcscmp(pwcs1, pwcs3);
    /*  retval has the result  */
}
```

マルチバイト文字とワイド文字のストリング比較サブルーチン

strcmp サブルーチンと **strncmp** サブルーチンは、 2 つのマルチバイト・ストリングが同一かどうかを判別します。 2 つのストリング間の字句の相違内容をアプリケーションが認識する必要がある場合は、マルチバイトおよびワイド文字ストリング比較サブルーチンを使用してください。

以下の各国語サポート・サブルーチンは、ワイド文字ストリングを比較します。

サブルーチン	説明
wcscmp	2 つのワイド文字ストリングを比較します。
wcsncmp	特定の数のワイド文字ストリングを比較します。

例:

次の例では、 **wcscmp** サブルーチンを使用して、 2 つのワイド文字ストリングを比較します。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
    int retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  pwcs1 and pwcs2 point to two wide character
```

```
    **  strings to compare.
    */
    retval = wcscmp(pwcs1, pwcs2);
    /*  pwcs1 contains a copy of the wide character string
    **  in pwcs2
    */
}
```

ワイド文字ストリング変換サブルーチン

以下の各国語サポート・サブルーチンは、ワイド文字ストリングを倍精度、長整数、および符号なし長整数に変換します。

サブルーチン	説明
wcstod	ワイド文字ストリングを倍精度の浮動小数点に変換します。
wcstol	ワイド文字ストリングを符号付きの長整数に変換します。
wcstoul	ワイド文字ストリングを符号なしの長整数に変換します。

wcstod、**wcstoul**、または **wcstol** サブルーチンを呼び出す場合は、その前に **errno** グローバル変数を 0 に設定しなければなりません。設定すると、これらのサブルーチンの呼び出し時に生じたエラーを正しく処理できるようになります。

例:

次の例では、 **wcstod** サブルーチンを使用して、ワイド文字ストリングの文字を倍精度の浮動小数点に変換します。

```
#include <stdlib.h>
#include <locale.h>
#include <errno.h>

extern int errno;

main()
{
    wchar_t  *pwcs, *endptr;
    double   retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  Let pwcs point to a wide character null terminated
    **  string containing a floating point value.
    */
    errno = 0;   /*  set errno to  zero  */
    retval = wcstod(pwcs, &endptr);

    if(errno != 0){
        /*  errno has changed, so error has occurred */

        if(errno == ERANGE){
            /*  correct value is outside range of
            **  representable values. Case of overflow
            **  error
            */

             if((retval == HUGE_VAL) ||
                (retval == -HUGE_VAL)){
                /*  Error case. Handle accordingly.  */
            }else if(retval == 0){
                /*  correct value causes underflow   */
                /*  Handle appropriately             */
```

```
            }
        }
    }
    /*  retval contains the double.  */
}
```

次の例では、 **wcstol** サブルーチンを使用して、ワイド文字ストリングの文字を符号付きの長整数に変換します。

```
#include <stdlib.h>
#include <locale.h>
#include <errno.h>
#include <stdio.h>

extern int errno;

main()
{
    wchar_t   *pwcs, *endptr;
    long int  retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  Let pwcs point to a wide character null terminated
    **  string containing a signed long integer value.
    */
    errno = 0;  /*  set errno to  zero  */
    retval = wcstol(pwcs, &endptr, 0);

    if(errno != 0){
        /*  errno has changed, so error has occurred */

        if(errno == ERANGE){
            /*  correct value is outside range of
            **  representable values. Case of overflow
            **  error
            */

             if((retval == LONG_MAX) || (retval == LONG_MIN)){
                /*  Error case. Handle accordingly.     */
            }else if(errno == EINVAL){
                /*  The value of base is not supported  */
                /*  Handle appropriately                */
            }
        }
    }
    /*  retval contains the long integer.  */
}
```

次の例では、 **wcstoul** サブルーチンを使用して、ワイド文字ストリングの文字を符号なしの長整数に変換します。

```
#include <stdlib.h>
#include <locale.h>
#include <errno.h>

extern int errno;

main()
{
    wchar_t    *pwcs, *endptr;
    unsigned long int  retval;

    (void)setlocale(LC_ALL, "");

    /*
```

```
    **  Let pwcs point to a wide character null terminated
    **  string containing an unsigned long integer value.
    */
    errno = 0;   /*  set errno to  zero  */
    retval = wcstoul(pwcs, &endptr, 0);

    if(errno != 0){
        /*  error has occurred */
        if(retval == ULONG_MAX || errno == ERANGE){
            /*
            **  Correct value is outside of
            **  representable value. Handle appropriately
            */
        }else if(errno == EINVAL){
            /*  The value of base is not representable  */
            /*  Handle appropriately                    */
        }
    }
    /*  retval contains the unsigned long integer.  */
}
```

ワイド文字ストリングのコピー・サブルーチン

以下の各国語サポート・サブルーチンは、ワイド文字ストリングをコピーします。

サブルーチン	説明
wcscpy	ワイド文字ストリングを、別のワイド文字ストリングにコピーします。
wcsncpy	特定の数の文字を、ワイド文字ストリング間でコピーします。
wcscat	ワイド文字ストリングを、別のワイド文字ストリングに追加します。
wcsncat	特定の数の文字を、ワイド文字ストリング間で追加します。

例:

次の例では、 **wcscpy** サブルーチンを使用して、ワイド文字ストリングをワイド文字のアレイにコピーします。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
    size_t  n;

    (void)setlocale(LC_ALL, "");
    /*
    **  Allocate the required wide character array.
    */
    pwcs1 = (wchar_t *)malloc( (wcslen(pwcs2) +1)*sizeof(wchar_t));
    wcscpy(pwcs1, pwcs2);
    /*
    **  pwcs1 contains a copy of the wide character string in pwcs2
    */
}
```

ワイド文字ストリング検索サブルーチン

以下の各国語サポート・サブルーチンは、ワイド文字ストリングの検索に使用します。

サブルーチン	説明
wcschr	ワイド文字ストリング中で最初に現れるワイド文字を検索します。
wcsrchr	ワイド文字ストリング中で最後に現れるワイド文字を検索します。
wcspbrk	ワイド文字ストリング中で最初に現れる複数のワイド文字を検索します。
wcsspn	ワイド文字ストリング中の初期セグメント中のワイド文字の数を判別します。
wcscspn	ワイド文字ストリングを検索します。
wcswcs	ワイド文字ストリング中で最初に現れる別のワイド文字ストリングを検索します。
wcstok	ワイド文字ストリングを、一連の独立したワイド文字ストリングに分けます。

例:

次の例では、 **wcschr** サブルーチンを使用して、ワイド文字ストリング中で最初に現れるワイド文字を見付けます。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, wc, *pws;
    int     retval;

    (void)setlocale(LC_ALL, "");

    /*
    **  Let pwcs1 point to a wide character null terminated string.
    **  Let wc point to the wide character to search for.
    **
    */
    pws = wcschr(pwcs1, wc);
    if (pws == (wchar_t )NULL ){
        /*  wc does not occur in pwcs1  */
    }else{
        /*  pws points to the location where wc is found  */
    }
}
```

次の例では、 **wcsrchr** サブルーチンを使用して、ワイド文字ストリング中で最後に現れるワイド文字を見付けます。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, wc, *pws;
    int     retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  Let pwcs1 point to a wide character null terminated string.
    **  Let wc point to the wide character to search for.
    **
    */
    pws = wcsrchr(pwcs1, wc);
    if (pws == (wchar_t )NULL ){
        /*  wc does not occur in pwcs1  */
```

```
    }else{
        /*  pws points to the location where wc is found  */
    }
}
```

次の例では、 **wcspbrk** サブルーチンを使用して、ワイド文字ストリング中で最初に現れる複数のワイド文字を見付けます。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2, *pws;

    (void)setlocale(LC_ALL, "");

    /*
    **  Let pwcs1 point to a wide character null terminated string.
    **  Let pwcs2 be initialized to the wide character string
    **  that contains wide characters to search for.
    */
    pws = wcspbrk(pwcs1, pwcs2);

    if (pws == (wchar_t )NULL ){
        /* No wide character from pwcs2 is found in pwcs1    */
    }else{
        /* pws points to the location where a match is found */
    }
}
```

次の例では、 **wcsspn** サブルーチンを使用して、ワイド文字ストリング・セグメントの初期セグメント中のワイド文字の数を判別します。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
    size_t count;

    (void)setlocale(LC_ALL, "");
    /*
    **  Let pwcs1 point to a wide character null terminated string.
    **  Let pwcs2 be initialized to the wide character string
    **  that contains wide characters to search for.
    */
    count = wcsspn(pwcs1, pwcs2);
    /*
    **  count contains the length of the segment.
    */
 }
```

次の例では、 wcscspn サブルーチンを使用して、ワイド文字ストリング・セグメント中に入っていないワイド文字の数を判別します。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
```

```
    size_t count;

    (void)setlocale(LC_ALL, "");

    /*
    **  Let pwcs1 point to a wide character null terminated string.
    **  Let pwcs2 be initialized to the wide character string
    **  that contains wide characters to search for.
    */
    count = wcscspn(pwcs1, pwcs2);
    /*
    **  count contains the length of the segment consisting
    **  of characters not in pwcs2.
    */
}
```

次の例では、 **wcswcs** サブルーチンを使用して、ワイド文字ストリング中で最初に現れる別のワイド文字ストリングを見付けます。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2, *pws;

    (void)setlocale(LC_ALL, "");
    /*
    **  Let pwcs1 point to a wide character null terminated string.
    **  Let pwcs2 be initialized to the wide character string
    **  that contains wide characters sequence to locate.
    */
    pws = wcswcs(pwcs1, pwcs2);
    if (pws == (wchar_t)NULL){
       /*  wide character sequence pwcs2 is not found in pwcs1 */
    }else{
        /*
        **  pws points to the first occurrence of the sequence
        **  specified by pwcs2 in pwcs1.
        */
    }
}
```

次の例では、 **wcstok** サブルーチンを使用して、ワイド文字ストリングをトークン化します。

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1 = L"?a???b,,,#c";
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");
    pwcs = wcstok(pwcs1, L"?");
    /*  pws points to the token:  L"a"  */
    pwcs = wcstok((wchar_t *)NULL, L",");
    /*  pws points to the token:  L"??b"  */
    pwcs = wcstok((wchar_t *)NULL, L"#,");
    /*  pws points to the token:  L"c"  */
}
```

ワイド文字入出力サブルーチン

各国語サポートには、定様式と不定形式の両方の入出力に関するサブルーチンが備えられています。

定様式のワイド文字の入出力:

printf サブルーチンと **scanf** サブルーチンを使用して、ワイド文字を形式設定できます。 **printf** サブルーチンと **scanf** サブルーチンには、ワイド文字の処理に関する追加の形式指定子として %C と %S の 2 つがあります。 %C 形式指定子を使用するとワイド文字の入出力が可能になり、 %S 形式指定子を使用するとワイド文字ストリングの入出力が可能になります。 これらに似たものとして、%c および %s 形式指定子があります。 %c および %s を使用すると、マルチバイトの文字とストリングの入出力を行えます。

マルチバイト・サブルーチンは、マルチバイト・アレイを受け入れ、マルチバイト・アレイを出力します。マルチバイト・サブルーチンからのマルチバイト出力をワイド文字ストリングに変換するには、**mbstowcs** サブルーチンを使用してください。

不定形式のワイド文字の入出力:

プログラムがマルチバイト・コード・セット中の文字を、コード・セットから独立して入出力する必要がある場合に、不定形式ワイド文字入出力サブルーチンを使用します。 例えば、 **fgetwc** サブルーチンか **getwc** サブルーチンを使用して、マルチバイト文字を入力します。 プログラムで **getc** サブルーチンを使用してマルチバイト文字を入力する場合、そのプログラムがマルチバイト文字中のバイト当たり 1 回ずつ **getc** サブルーチンを呼び出すようにしなければなりません。

ワイド文字入力サブルーチンは、ストリームからマルチバイト文字を読み取って、ワイド文字に変換します。 この種のサブルーチンが **mbtowc** サブルーチンや **mbstowcs** サブルーチンを呼び出した場合と同様の変換が行われます。

ワイド文字出力サブルーチンは、ワイド文字をマルチバイト文字に変換して、ストリームに書き込みます。この種のサブルーチンが **wctomb** および **wcstombs** サブルーチンを呼び出した場合と同様の変換が行われます。

ワイド文字入出力サブルーチンの動作は、現行ロケールの **LC_CTYPE** カテゴリーに制御されます。

ファイル全体の読み取りと処理:

プログラムが、ワイド文字コード形式で処理しなければならないファイルの全体を検索する必要がある場合は、以下のいずれかの方法を使用してください。

- マルチバイト文字の場合は、**read** または **fread** のいずれかのサブルーチンを使用して、テキスト・データ・ブロックをバッファーに変換します。 このバッファーで、**mbtowc** サブルーチンを用いて、一度に 1 文字ずつ変換します。 マルチバイト文字がブロック境界を越えるという特殊なケースを処理します。マルチバイト・コード・セットの場合は、このバッファーで **mbstowcs** サブルーチンを使用しないでください。 無効マルチバイト文字シーケンス、あるいは部分マルチバイト文字シーケンスの場合は、**mbstowcs** サブルーチンは、データをどこまで正常に変換したかは示さずに、-1 を戻します。 単一バイト・コード・セットでは部分バイト・シーケンスの問題は発生しないので、単一バイト・コード・セットには **mbstowcs** サブルーチンを使用することができます。
- **fgetws** サブルーチンを使用してファイルから 1 行を入手します。戻されたワイド文字ストリングにワイド文字 <new-line> が入っていれば、完全な 1 行が入手されます。 <new-line> ワイド文字がない場合は、その行が予期したよりも長いものであり、行全体を入手するには、さらに **fgetws** サブルーチンを呼び出す必要があります。 プログラムが一度に 1 行を効率的に処理できる場合は、この方法をお勧めします。
- 一度に 1 行入手するのに、 **fgets** サブルーチンを使用してマルチバイト・ファイルを読み取る場合には、結果としてマルチバイト文字が分割されることがあります。この状態は、 **read** サブルーチンがマルチバイト文字を連続の読み取りで分割する場合と同じように処理してください。 入力行の長さが定め

られた制限を超えないという保証がある場合は、そのサイズ (ヌルのために 1 を加える) のバッファーを使用することによって、マルチバイト文字が分割されないようにすることができます。 プログラムが一度に 1 行を効率的に処理できる場合は、この方法を使用することができます。 バッファー内でバイトが分割される可能性があるため、マルチバイト文字には **fgets** サブルーチンより **fgetws** サブルーチンを優先して使用してください。

- 一度に 1 つのワイド文字コードを読み取る場合は、ファイルに **fgetwc** サブルーチンを使用します。ファイルが大きい場合は、関数呼び出しのオーバーヘッドが大きくなり、この方法の価値は減少します。

これらの方法のどれを使用するかの決定は、プログラム単位に行う必要があります。 **fgetsw** サブルーチンのオプションは、最適なパフォーマンスを得られる可能性があり、プログラムが特殊なケースを処理する必要がないため、お勧めします。

入力サブルーチン*:*

ファイルの終わり (EOF) マーカーだけでなく、ワイド文字コード値を表すために **wint_t** データ型が必要です。 例えば、ワイド文字コード値を戻す、 **fgetwc** サブルーチンのケースを考えます。

サブルーチン戻り値の宣言	説明
wchar_t fgetwc();	**wchar_t** データ型が **char** 値として定義されていると、 y ウムラウト記号は ISO8859-1 コード・セットのファイルの終わり (EOF) マーカーと区別がつきません。 0xFF コード・ポイントは、有効な文字です (y ウムラウト)。 したがって、戻り値を **wchar_t** データ型にすることはできません。 EOF マーカーと、コード・セット内のすべてのコード・ポイントの両方を保持することができるデータ型が必要です。
int fgetwc();	マシンによっては、 **int** データ型が 16 ビットに定義される場合もあります。 **wchar_t** データ型が 16 ビットより大きい場合、 **int** 値は戻り値をすべて表すことはできません。

したがって、**fgetwc** サブルーチン戻り値を表すには、 **wint_t** データ型が必要です。 **wint_t** データ型は、 **wchar.h** ファイルに定義されています。

サブルーチン戻り値	説明
fgetwc	ストリームから次のワイド文字を入手する。
fgetws	ストリームからワイド文字ストリングを入手する。
getwc	ストリームから次のワイド文字を入手する。
getwchar	標準入力から次のワイド文字を入手する。
getws	標準入力からワイド文字ストリングを入手する。
ungetwc	ワイド文字をストリームにプッシュする。

出力サブルーチン*:*

以下のサブルーチンは、ワイド文字出力に使用します。

サブルーチン	説明
fputwc	ワイド文字を出力ストリームに書き出す。
fputws	ワイド文字ストリングを出力ストリームに書き出す。
putwc	ワイド文字を出力ストリームに書き出す。
putwchar	ワイド文字を標準出力に書き出す。
putws	ワイド文字ストリングを標準出力に書き出す。

例:

次の例では、 **fgetwc** サブルーチンを使用して、ファイルからワイド文字コードを読み取ります。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wint_t  retval;
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    /*
    **  Open a stream.
    */
    fp = fopen("file", "r");

    /*
    **  Error Handling if fopen was not successful.
    */
    if(fp == NULL){
        /*  Error handler  */
    }else{
        /*
        **  pwcs points to a wide character buffer of BUFSIZ.
        */
        while((retval = fgetwc(fp)) != WEOF){
            *pwcs++ = (wchar_t)retval;
               /*  break when buffer is full  */
        }
    }
    /*  Process the wide characters in the buffer  */
}
```

次の例では、 **getwchar** サブルーチンを使用して、標準入力からワイド文字を読み取ります。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

 main()
{
    wint_t  retval;
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    index = 0;
    while((retval = getwchar()) != WEOF){
        /*  pwcs points to a wide character buffer of BUFSIZ.  */
        *pwcs++ = (wchar_t)retval;
```

```
        /*  break on buffer full  */
    }
    /*  Process the wide characters in the buffer  */
}
```

次の例では、 **ungetwc** サブルーチンを使用して、ワイド文字を入力ストリームにプッシュします。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wint_t  retval;
    FILE    *fp;

    (void)setlocale(LC_ALL, "");
    /*
    **  Open a stream.
    */
    fp = fopen("file", "r");

    /*
    **  Error Handling if fopen was not successful.
    */
    if(fp == NULL){
        /*  Error handler  */

    else{
        retval = fgetwc(fp);
        if(retval != WEOF){
            /*
            **  Peek at the character and return it to the stream.
            */
            retval = ungetwc(retval, fp);
            if(retval == EOF){
                /*  Error on ungetwc  */
            }
        }
    }
}
```

次の例では、 **fgetws** サブルーチンを使用して、ファイルから一度に 1 行ずつ読み取ります。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    /*
    **  Open a stream.
    */
    fp = fopen("file", "r");

    /*
    **  Error Handling if fopen was not successful.
    */
    if(fp == NULL){
        /*  Error handler  */
    }else{
        /*  pwcs points to wide character buffer of BUFSIZ.  */
```

```
        while(fgetws(pwcs, BUFSIZ, fp) != (wchar_t *)NULL){
            /*
            **  pwcs contains wide characters with null
            **  termination.
            */
        }
    }
}
```

次の例では、 **fputwc** サブルーチンを使用して、ワイド文字を出力ストリームに書き出します。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    int     index, len;
    wint_t  retval;
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    /*
    **  Open a stream.
    */
    fp = fopen("file", "w");

    /*
    **  Error Handling if fopen was not successful.
    */
    if(fp == NULL){
        /*  Error handler  */
    }else{
        /*  Let len indicate number of wide chars to output.
        **  pwcs points to a wide character buffer of BUFSIZ.
        */
        for(index=0; index < len; index++){
            retval = fputwc(*pwcs++, fp);
            if(retval == WEOF)
                break;  /*  write error occurred  */
                        /*  errno is set to indicate the error. */
        }
    }
}
```

次の例では、 **fputws** サブルーチンを使用して、ファイルにワイド文字ストリングを書き込みます。

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    int     retval;
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    /*
    **  Open a stream.
    */
    fp = fopen("file", "w");

    /*
```

```
    **  Error Handling if fopen was not successful.
    */
    if(fp == NULL){
        /*  Error handler  */

    }else{
        /*
        **  pwcs points to a wide character string
        **  to output to fp.
        */
        retval = fputws(pwcs, fp);
        if(retval == -1){
            /*  Write error occurred                */
            /*  errno is set to indicate the error  */
        }
    }
}
```

ワイド文字定数の処理

L 定数は ASCII 文字のみに使用してください。 ASCII 文字の場合、**L** 定数値は、数値としては文字のコード・ポイント値と同じです。 例えば、 **L'a'** は a と同じです。 **L** 定数は、割り当てが目的で、ASCII 文字の **wchar_t** 値を入手します。 ワイド文字定数は、**L** 指定子によって導入されます。 例えば、次のようになります。

```
wchar_t wc = L'x' ;
```

x の文字に対応するワイド文字コードが、 wc に保管されています。 C コンパイラーは、 **mbtowc** または **mbstowcs** サブルーチンを適宜用いて、 x の文字を変換します。 このワイド文字への変換は、コンパイル時の現行ロケールの設定に基づいています。 ASCII 文字は、サポートされているすべてのコード・セットの一部であり、すべての ASCII 文字のワイド文字表現は、すべてのロケールにおいて同じであるため、 L'x' は、すべてのコード・セットにおいて同じ値になります。 しかし、x の文字が非 ASCII であると、プログラムは、コンパイル時に使用されたのと異なるコード・セットで実行されると作動しないことがあります。 この制限は、ワイド文字定数表現を用いる切り替えステートメントを使用するプログラムに影響を与える場合があります。

wchar.h ヘッダー・ファイル

wchar.h ヘッダー・ファイルは、マルチバイト文字サブルーチンおよびワイド文字サブルーチンでプログラミングを行う場合に必要な情報を宣言します。 **wchar.h** ヘッダー・ファイルは、ワイド文字をテストするためのいくつかの関数に加えて、 **wchar_t**、**wctype_t**、および **wint_t** データ型を宣言します。 ワイド文字としてインプリメントされている文字の数は、基本文字の文字数を超えるため、すべてのワイド文字を、基本文字に使用する既存のクラスに分類することは不可能です。 したがって、ロケールによっては固有の追加クラスを定義する方法を用意する必要があります。 これらのサブルーチンのアクションは、現行ロケールの影響を受けます。

wchar.h ヘッダー・ファイルは、ワイド文字ストリング (すなわち、 **wchar_t** データ型アレイ) を操作するサブルーチンも宣言します。 配列の長さは、常にアレイ内の **wchar_t** エレメントの数によって決定されます。 アレイはヌル・ワイド文字コードで終了します。 **wchar_t** データ型アレイを指すポインター、またはボイド・アレイは、常にアレイの最初のエレメントを指します。

> 注**:** アレイ内の **wchar_t** エレメントの数が、定義されたアレイの長さを超えると、予期しない結果が生じる場合があります。

国際化正規表現サブルーチン

国際化正規表現を含むプログラムは、**regcomp**、**regexec**、**regerror**、**regfree**、および **fnmatch** サブルーチンを使用することができます。

以下のサブルーチンは、国際化正規表現の場合に使用することができます。

regcomp
指定された基本正規表現または拡張正規表現を、実行可能ストリングにコンパイルする。

regexec
ヌル終了ストリングと、前に **regcomp** サブルーチンの呼び出しによってコンパイルされていなければならないコンパイル済み基本正規表現または拡張正規表現を比較する。

regerror
regcomp および **regexec** サブルーチンが戻したエラー・コードから、印刷可能なストリングへのマッピングを行う。

regfree
コンパイル済み基本正規表現または拡張正規表現に関連する、**regcomp** サブルーチンによって割り当てられたメモリーを解放する。 この式は、**regfree** サブルーチンに指定された後は、コンパイル済み基本正規表現または拡張正規表現としては扱われなくなります。

fnmatch
指定されたストリングを検査して、指定されたパターンに一致しているか調べる。辞書を読み取るアプリケーションで **fnmatch** サブルーチンを使用して、与えられたパターンに一致しているエントリーを見つけることができます。 また、**fnmatch** サブルーチンを使用すると、パス名をパターンに突き合わせることもできます。

例

次の例では、国際化正規表現をコンパイルし、このコンパイルされた表現を使用してストリングを突き合わせます。最初のパターンには一致が見つかりますが、2 番目のパターンには一致は見つかりません。

```
#include                <locale.h>
#include                <regex.h>

#define                 BUFSIZE    256


main()
{
        char    *p;

        char    *pattern[] = {
                "hello[0-9]*",
                "1234"
        };


        char     *string = "this is a test string hello112 and this is test";
                /* This is the source string for matching */

        int     retval;
        regex_t   re;
        char    buf[BUFSIZE];

        int     i;

        setlocale(LC_ALL, "");

        for(i = 0;i <2; i++){
```

```
                retval = match(string, pattern[i], &re);
                if(retval == 0){
                        printf("Match found ¥n");
                }else{
                        regerror(retval, &re, buf, BUFSIZE);
                        printf("error = %s¥n", buf);
                }
        }
        regfree( &re);
}


int match(char *string, char *pattern, regex_t *re)
{
        int     status;

        if((status=regcomp( re, pattern, REG_EXTENDED))!= 0)
                return(status);
        status = regexec( re, string, 0, NULL, 0);
        return(status);
}
```

次の例では、1 行のなかの、パターンに一致するすべてのサブストリングを検出します。 11 と 2001 の数が突き合わせられます。 桁が一致するたびに 1 つの一致と数えられます。 ストリングに与えられた 6 桁に対応して、その種の一致が 6 つあります。

```
#include         <locale.h>
#include                <regex.h>

#define                BUFSIZE   256


main()
{
        char    *p;

        char    *pattern = "[0-9]";
        char    *string = "Today is 11 Feb 2001 ";

        int     retval;
        regex_t re;
        char    buf[BUFSIZE];
        regmatch_t pmatch[100];
        int      status;
        char     *ps;

        int      eflag;

        setlocale(LC_ALL, "");

        /* Compile the pattern */

        if((status = regcomp( &re, pattern, REG_EXTENDED))!= 0){
                regerror(status, &re, buf, 120);
                exit(2);
        }

        ps = string;
        printf("String to match=%s¥n", ps);
        eflag = 0;

        /* extract all the matches */
        while( status = regexec( &re, ps, 1, pmatch, eflag)== 0){
                printf("match found at: %d, string=%s¥n",
                        pmatch[0].rm_so, ps +pmatch[0].rm_so);
                ps += pmatch[0].rm_eo;
```

```
                printf("¥nNEXTString to match=%s¥n", ps);
                eflag = REG_NOTBOL;

        }
        regfree( &re);
}
```

次の例では、**fnmatch** サブルーチンを使用して、ディレクトリーを読み取り、ファイル名をパターンに突き合わせます。

```
#include                <locale.h>
#include                <fnmatch.h>
#include                <sys/dir.h>

main(int argc, char *argv[] )
{
        char    *pattern;
        DIR     *dir;
        struct dirent    *entry;
        int     ret;


                setlocale(LC_ALL, "");

        dir = opendir(".");

        pattern = argv[1];

        if(dir != NULL){
                while( (entry = readdir(dir)) != NULL){
                        ret = fnmatch(pattern, entry->d_name,
                                FNM_PATHNAME|FNM_PERIOD);
                        if(ret == 0){
                                printf("%s¥n", entry->d_name);
                        }else if(ret == FNM_NOMATCH){
                                continue ;
                        }else{
                                printf("error file=%s¥n",
                                        entry->d_name);
                        }
                }
                closedir(dir);
        }
}
```

各国語サポートのコード・セット

AIX のグローバリゼーションは、すべてのコード・セットは任意の数の文字セットに分けられるという仮定に基づいています。

コード・セットを理解するためには、まず文字セットを理解する必要があります。 文字セットとは、文字を表すのに使用される値のエンコードに関係なく、 1 つ以上の言語の特定の必要に基づいて事前に定義された文字の集合です。 使用するコード・セットの選択は、ユーザーのデータ処理要件によって異なります。特定の文字セットは、別のエンコード・スキームを使用してエンコードできます。 例えば、ASCII 文字セットでは、英語にある文字群を定義しています。 日本工業規格 (JIS) 文字セットでは、日本語で使用される文字群を定義しています。英語と日本語のどちらの文字セットも、異なるコード・セットを使用してエンコードできます。

コード・ページ とは、コード・ページ仕様は 16 列×16 行の行列に基づく、という制限付きのコード・セットのようなものです。それぞれの列と行の交点では、コード化文字を定義します。

コード・セットを扱うときには、以下の点を考慮してください。

- 文字のサイズはすべて 8 ビット (1 バイト) であると考えないでください。文字は、1、2、3、4 またはそれ以上のバイトになる可能性があります。
- コード・セットのエンコードを決め付けないでください。
- 移植性に影響する可能性があるため、コード・セット、ロケール、またはフォントの名前をハードコーディングしないでください。

以下のコード・セットがサポートされています。

- 業界標準のコード・セットがサポートされています。 **ISO8859** ファミリーのコード・セットでは、以下を含む一定の範囲の単一バイト・コード・セットがサポートされています。
  - Latin-1
  - Latin-2
  - Latin-4
  - キリル文字
  - アラビア語
  - ギリシャ語
  - ヘブライ語
  - トルコ語

  以下の業界標準コード・セットが使用可能です。
  - IBM-eucJP コード・セットは、日本語ロケールをサポートするのに使用される業界標準コード・セットです。
  - IBM-eucKR コード・セットは、ハングルをサポートするのに使用される業界標準コード・セットです。
  - IBM-eucTW コード・セットは、中国語 (繁体字) をサポートするのに使用される業界標準コード・セットです。
  - IBM-eucCN コード・セットは、中国語 (簡体字) を使用する国々をサポートするのに使用される業界標準コード・セットです。
  - UTF-8 コード・セットは、複数の言語 (中国語 (簡体字)、中国語 (繁体字)、および日本語と韓国語で使用される漢字) を同時にサポートするのに使用される Universal Transformation Format of Unicode/ISO10646 のことです。
- **ISO8859-15** 標準コード・セットは、現在、西ヨーロッパ・ロケール、米国、およびカナダで使用されている既存の ISO8859-1 コード・セットに代わる規格です。 別のコード・セットが必要になったのは、ユーロ通貨単位が導入されたため、ヨーロッパ諸国でユーロを使用した商取引ができるようにする必要があったからです。 さらに、ISO8859-15 には、フランス語およびフィンランド語用に、7 つの文字が追加されています。
- パーソナル・コンピューター (PC) ベースのコード・セット IBM-856、IBM-943、および IBM-1046 もサポートされています。IBM-856 は、ヘブライ語をサポートするときに使用される単一バイト・コード・セットです。 IBM-943 は、日本語ロケールをサポートするときに使用されるマルチバイト・コード・セットです。IBM-1046 は、アラビア語諸国をサポートするときに使用される単一バイト・コード・セットです。
- **IBM-1129** は、ベトナム語をサポートするときに使用される単一バイト・コード・セットです。
- **TIS-620** は、タイ語をサポートするときに使用される単一バイト・コード・セットです。
- **IBM-1124** は、ウクライナ語をサポートするときに使用される単一バイト・コード・セットです。

- AIX でサポートされているすべて の言語と地域で、**UTF-8** コード・セットによって、ユニコードが完全にサポートされています。 UTF-8 コード・セットは、複数の言語を同時にサポートするのに使用される Universal Transformation Format of Unicode/ISO10646 のことです。 UTF-8 コード・セットには、複数の言語およびアルファベットを処理しなければならない環境で使用するための、完全なソリューションが備えられています。また Unicode/UTF-8 コード・セットでは、共通のヨーロッパ通貨 (ユーロ) も完全にサポートされています。
- **IBM-1252** コード・セットは、ユーロ通貨記号を含む単一バイト・コード・セット環境を必要とするユーザーのための互換オプションとして、サポートされています。 IBM-1252 コード・セットの構造は、業界標準のコード・セット ISO8859-1 と同じです。ただし、0x80 から 0x9F までの ISO 制御文字の範囲内に、別の図形文字が追加されているところは異なります。 IBM-1252 コード・セットでは、ユーロ通貨記号は 16 進数の valTXx80 にあります。

関連概念:

55 ページの『ASCII 文字』
ASCII は、128 のコード・ポイント (0x00 から 0x7F) を含むコード・セットです。 ASCII 文字セットには、制御文字、句読記号、数字、および大文字と小文字の英語アルファベットが含まれます。 いくつかの 8 ビットのコード・セットが、適切なサブセットとして ASCII と一体になっています。 しかし、本書では、ASCII は 7 ビットのコード・セットのみを指すものとします。 このことを強調するため、7 ビット ASCII と呼ばれます。 7 ビット ASCII コード・セットは、サポートされているすべてのコード・セットのサブセットで、ポータブル文字セット と呼ばれます。

93 ページの『コンバーターのリスト』
コンバーターは、データをあるコード・セットから別のコード・セットに変更します。 次のセクションでは、iconv ライブラリーでサポートされているコンバーター群がリストされています。

単一バイトとマルチバイトのコード・セット

単一バイト・エンコード方式は、文字数が多くないので、英語の文字を表すのに十分です。 日本語や中国語など、より大きなアルファベットをサポートするには、マルチバイト・エンコードを含む別のコード・セットが必要になります。サポートされている単一バイト・コード・セットとマルチバイト・コード・セットにはすべて、単一バイト ASCII 文字セットが含まれます。 したがって、マルチバイト・コード・セットを扱うプログラムは、 1 バイト以上の文字エンコードを扱わなければなりません。

単一バイト・コード・セットの一例として、 ISO 8859 ファミリーのコード・セットがあります。 マルチバイト文字セットの例としては、IBM-eucJP および IBM-943 コード・セットがあります。 単一バイト・コード・セットは最大で 256 文字であり、マルチバイト・コード・セットは 256 を超えます (理論的な制限はありません)。

固有のコード・ポイント範囲

マルチバイト文字のいずれかのバイトに 0x00 から 0x3F を含むコード・セットはサポートされていません。 このグループのコード・ポイントのことを、固有のコード・ポイント範囲 といいます。

これらのコード・ポイントは常に、7 ビット ASCII に指定されたのと同じ文字を参照します。 これは、サポートされるすべてのコード・セットを管理する特殊な属性です。 固有のコード・ポイント範囲に含まれる ASCII 文字 (55 ページの『ASCII 文字』) は、固有のコード・ポイント範囲の文字をリストします。

データ表現

いくつかの文字のエンコードでは複数のバイトを必要とするため、データがファイル内で作成される場合や、コンピューターと入出力装置間で転送されるときには、 1 つの文字を 1 バイト以上で表現できます。このデータの外部表現のことを、ファイル・コード、または文字の マルチバイト文字コード 表現といいます。

そのような文字ストリングを処理するためには、ファイル・コードを一律の表現に変換するとより効率的です。 変換後の形式は、文字の内部処理に使用できます。 このデータの内部表現のことを、処理コード、または文字の ワイド文字コード 表現といいます。 マルチバイト文字コードとワイド文字コードを理解することは、グローバリゼーション・ストラテジーの全体に必要なことです。

マルチバイト文字コードのデータ表現

マルチバイト文字コードとは、そのデータがキーボードからの文字入力なのかディスク上のファイルなのかに関係なく、データの外部表現のことです。 同じコード・セット内では、文字のマルチバイト・コードを表すバイト数はさまざまです。 文字処理用の各国語サポート機能を使用して、コード・セットの独立性を確実にする必要があります。

例えば、あるコード・セットでは、次の文字エンコードを指定できます。

```
C  = 0x43
*  = 0x81 0x43
*C = 0x81 0x43& 0x43
```

マルチバイト文字を含まない C を検索するプログラムは、*C ストリングの 2 番目のバイトを検出し、実際に * (アスタリスク) 文字の 2 番目のバイトを検出した場合、C を検出したと認識します。

ワイド文字のデータ表現

ワイド文字コードは、システム内部でマルチバイト文字をより効率的に処理できるようにするために開発されました。マルチバイト文字表現は、内部ではすべての文字が同じ長さになるように、一律の内部表現 (ワイド文字コード) に変換されます。 この内部形式を使用すると、文字の処理をコード・セットから独立した方法で行えます。 ワイド文字コードは、この文字の内部表現を参照します。

wchar_t データ型は、ある文字のワイド文字コードを表現するために使用されます。 **wchar_t** データ型のサイズは、インプリメンテーションごとに異なります。 これは **typedef** 定義で、**ctype.h**、**stddef.h**、および **stdlib.h** ファイル内に存在します。プログラムでは、**wchar_t** データ型のために特定のサイズを想定してはならず、**wchar_t** データ型のために異なるサイズを使用するインプリメンテーションの下でプログラムを実行できるようにします。

AIX オペレーティング・システムでは、**wchar_t** データ型は、64 ビット環境では 32 ビット、32 ビット環境では 16 ビットです。ロケール方式は標準化されており、ほとんどのロケールでは、特定の文字の **wchar_t** に保管された値は、必ずその Unicode データ値になります。AIX 上でのみ実行されるよう意図されたアプリケーションの場合、基礎コード・セットが不明であっても、特定のアプリケーションは一貫性のある方法で **wchar_t** データ型を処理することができます。ロケールはすべて、ワイド文字コード値 (処理コード) に Unicode を使用します。ただし、IBM-eucTW コード・セットは例外です。 IBM-eucTW コード・セット (LANG =**zh_TW**) には、 Unicode 規格には含まれていない多くの文字が含まれています。そのため、これらの文字を Unicode ワイド文字値で表現することは不可能です。中国語 (繁体字) 向けに Unicode ベースの **wchar_t** データを持つことが必要なアプリケーションでは、代わりに **Zh_TW** ロケール (big5 コード・セット) を使用する必要があります。

char データ型が signed か unsigned かを想定しないでください。 それはプラットフォーム固有です。使用される特定のシステムが **char** を **signed** と定義する場合、完全な 8 ビット数量での比較は不正確な

結果になります。8 ビットはすべて、文字のエンコードに使用されるため、必要な場合には必ず、 **char** を **unsigned char** と宣言してください。 また、配列を索引付けするために **signed char** 値が使用される場合も、不正確な結果になる場合があります。プログラムを移植可能にするには、8 ビット文字を **unsigned char** と定義します。

文字の属性

それぞれの文字には、言語に依存した属性がいくつかあります。 これらの属性のことを、クラス属性 といいます。例えば、米国英語での小文字の a には、以下のような属性があります。

- 英字
- 16 進数字
- 印刷可能
- 小文字
- 図形

文字クラス属性は、**LC_CTYPE** カテゴリーで指定されます。

照合配列属性

文字配列すなわち照合は、文化に固有の文字配列を指します。 この配列は、コード・セットの文字の元の値に基づく配列とは異なります。

文字配列 すなわち 照合 は、文化に固有の文字配列を指します。 この配列は、コード・セットの文字の元の値に基づく配列とは異なります。 照合ベースの配列は、言語によって異なります。 文字配列は、LC_COLLATE カテゴリーによって指定されます。 照合エレメント という用語は、特定のロケールに特定の照合値を持つ 1 つ以上の文字を指します。 スペイン語の ll 文字は、複数文字による照合エレメントの一例です。

指定する任意の言語で文字を正しい配列にソートするため、指定どおりに文字をソートするために、各文字に重みが割り当てられます。しかし、文字のソート値およびコード・ポイント値は、必ずしも関連しているわけではありません。

すべての言語のストリングをソートするには、 1 組の重みだけでは十分ではありません。例えば、ドイツ語の単語 b<*a-umlaut*>ch と bane の場合、もし 1 組の重みだけが存在し、文字 a の重みが <*a-umlaut*> の重みよりも少なければ、bane は b<*a-umlaut*>ch の前にソートされます。 しかし、反対の結果が正しいものです。 この例の要件を満たすには、言語のそれぞれの文字に対して、 2 組の重み、すなわち 1 次の重みと 2 次の重みが指定されます。 文字 a と <*a-umlaut*> の場合、それらの 1 次の重みは同じですが、2 次の重みは異なります。 ドイツ語ロケールでは、a の 2 次の重みは、<*a-umlaut*> の 2 次の重みよりも少なくなります。

ソート・アルゴリズムでは、まず、各文字の 1 次の重みに基づき、 2 つのストリングを比較します。 1 次の重みの値が同じであれば、2 つのストリングは、2 次の重みに基づいてもう一度比較されます。 この例では、最初の 2 文字である ba と b<*a-umlaut*> の 1 次の重みは同じですが、その後に続く文字 (それぞれ c と *n*) の 1 次の重みは異なります。 この比較の結果、b<*a-umlaut*>ch は bane の前にソートされます。

ここでは、ストリングを照合するために 2 次の重みは使用されていません。しかし、ストリング bach と b<*a-umlaut*>ch の場合もそうですが、適切に配列するためには 2 次の重みを使用しなければなりません。1 次の重みの値を使用して比較する場合、これら 2 つのストリングは同等と見なされます。このつながり

を断つために、a と *<a-umlaut>* の 2 次の重みが使用されます。 a の 2 次の重みは、*<a-umlaut>* の 2 次の重みよりも少ないため、ストリング bach は、b*<a-umlaut>*ch より前にソートされます。

同じ 1 次の重みを持つ文字は、同じ 等価クラス に属します。 この例では、文字 a と *<a-umlaut>* は同じ等価クラスのメンバーであると考えられています。

ストリング照合では、ストリングのそれぞれのペアは、まず 1 次の重みに基づいて比較されます。 2 つのストリングが等しければ、2 次の重みに基づいてもう一度比較されます。 まだ等しければ、**sys/limits.h** ファイルで指定した **COLL_WEIGHTS_MAX** 照合の重みの制限で設定した制限まで、 3 次の重みに基づいてもう一度比較されます。

コード・セットの幅

コード・セットの幅 は、文字をファイル・コードとして表示するのに必要な最大バイト数を指します。 この情報は、**LC_CTYPE** カテゴリーによって指定されます。

コード・セットの表示幅

コード・セットの表示幅 は、端末で文字を表示するのに必要な最大列数を指します。 この情報は、**LC_CTYPE** カテゴリーによって指定されます。

ASCII 文字

ASCII は、128 のコード・ポイント (0x00 から 0x7F) を含むコード・セットです。 ASCII 文字セットには、制御文字、句読記号、数字、および大文字と小文字の英語アルファベットが含まれます。 いくつかの 8 ビットのコード・セットが、適切なサブセットとして ASCII と一体になっています。 しかし、本書では、ASCII は 7 ビットのコード・セットのみを指すものとします。 このことを強調するため、7 ビット ASCII と呼ばれます。 7 ビット ASCII コード・セットは、サポートされているすべてのコード・セットのサブセットで、ポータブル文字セット と呼ばれます。

関連概念:

50 ページの『各国語サポートのコード・セット』
AIX のグローバリゼーションは、すべてのコード・セットは任意の数の文字セットに分けられるという仮定に基づいています。

固有のコード・ポイント範囲内の ASCII 文字

次の表は、固有のコード・ポイント範囲に含まれる ASCII 文字をリストしています。 これらの文字は、0x00 から 0x3F の範囲内にあります。

表 2. 固有のコード・ポイント範囲の *ASCII* 文字

シンボル名	**16** 進値	絵文字	シンボル名	**16** 進値	絵文字
ヌル	00		スペース	20	ブランク
soh (ヘッダー開始)	01		感嘆符	21	!
stx (テキスト開始)	02		引用符	22	"
etx (テキスト終結)	03		番号記号	23	#
eot (伝送終了)	04		ドル記号	24	$
enq (問い合わせ)	05		パーセント	25	%
ack (応答)	06		アンパーサンド	26	&
アラート	07		アポストロフィ	27	'
バックスペース	08		左括弧	28	(
タブ	09		右括弧	29	)
改行	0A		アスタリスク	2A	*

表 2. 固有のコード・ポイント範囲の *ASCII* 文字 (続き)

シンボル名	**16** 進値	絵文字	シンボル名	**16** 進値	絵文字
垂直タブ	0B		正符号	2B	+
用紙送り	0C		コンマ	2C	,
復帰	0D		ハイフン	2D	-
so (シフトアウト)	0E		ピリオド	2E	.
si (シフトイン)	0F		スラッシュ	2F	/
dle (伝送制御拡張)	10		ゼロ	30	0
dc1	11		一	31	1
dc2	12		二	32	2
dc3	13		三	33	3
dc4	14		四	34	4
nak (否定応答)	15		五	35	5
syn (同期)	16		六	36	6
etb (伝送ブロック終結)	17		七	37	7
can (取り消し)	18		八	38	8
em (メディア終端)	19		九	39	9
sub (置き換え)	1A		コロン	3A	:
esc (エスケープ)	1B		セミコロン	3B	;
is1	1C		より小	3C	<
is2	1D		等号	3D	=
is3	1E		より大	3E	>
is4	1F		疑問符	3F	?

その他の ASCII 文字

次の表は、固有のコード・ポイント範囲に含まれない 7 ビット ASCII 文字をリストしています。 これらの文字は、0x40 から 0x7F の範囲内にあります。

表 3. 他の *ASCII* 文字

シンボル名	**16** 進値	絵文字	シンボル名	**16** 進値	絵文字
commercial-at (商用アットマーク)	40	@	抑音符号	60	`
A	41	A	a	61	a
B	42	B	b	62	b
C	43	C	c	63	c
D	44	D	d	64	d
E	45	E	e	65	e
F	46	F	f	66	f
G	47	G	g	67	g
H	48	H	h	68	h
I	49	I	i	69	i
J	4A	J	j	6A	j
K	4B	K	k	6B	k
L	4C	L	l	6C	l
M	4D	M	m	6D	m
N	4E	N	n	6E	n
O	4F	O	o	6F	o

表 3. 他の ASCII 文字 (続き)

シンボル名	16 進値	絵文字	シンボル名	16 進値	絵文字
P	50	P	p	70	p
Q	51	Q	q	71	q
R	52	R	r	72	r
S	53	S	s	73	s
T	54	T	t	74	t
U	55	U	u	75	u
V	56	V	v	76	v
W	57	W	w	77	w
X	58	X	x	78	x
Y	59	Y	y	79	y
Z	5A	Z	z	7A	z
左大括弧	5B	[	左中括弧	7B	{
バックスラッシュまたは円記号	5C	¥	縦線	7C	|
右大括弧	5D	]	右中括弧	7D	}
曲折アクセント記号	5E	^	波形記号	7E	~
下線	5F	_	削除	7F	

コード・セット戦略

システムの各ロケールでは、使用するコード・セットと、コード・セット内の文字の扱い方が定義されています。 システムには複数のロケールをインストールできるため、システム上で、異なるユーザーが複数のコード・セットを使用できます。 システムは異なるコード・セットを使用するロケールで構成できますが、システム・ユーティリティーはすべて、システムが 1 つのコード・セットの下で実行されているものと見なします。

ほとんどのコマンドは、ロケールで使用されている基礎となるコード・セットの内容を知りません。 コード・セットの内容は、コード・セットから独立したライブラリー・サブルーチン (グローバリゼーション・ライブラリー) によって隠されています。このサブルーチンは、コード・セットに依存するサブルーチンに情報を渡します。

多くのプログラムは ASCII に依存しているため、コード・セットにはすべて、適切なサブセットとして 7 ビット ASCII コード・セットが含まれています。 7 ビット ASCII コード・セットは、サポートされているすべてのコード・セットに共通なので、その文字は ポータブル文字セット と呼ばれることがあります。 7 ビット ASCII コード・セットは、ISO646 定義に基づくもので、文字、句読文字、数字 (0-9)、大文字および小文字の英語アルファベットを含みます。

コード・セットの構造

各コード・セットは、以下の基本領域に分けられます。

- グラフィック左 (GL): 列 0-7
- グラフィック右 (GR): 列 8-F

各コード・セットの最初の 2 つの列は、制御文字用に、国際標準化機構 (ISO) 規格で予約済みです。 C0 および C1 は、それぞれグラフィック左およびグラフィック右の領域用の制御文字を示すのに使用されています。

注: IBM PC コード・セットは、C1 制御域を使用して図形文字をエンコードします。

残りの 6 つの列は、図形文字をエンコードするのに使用されます。図形文字は、印刷可能文字であると考えられるのに対して、制御文字は、特別な機能を示すためにデバイスやアプリケーションによって使用されます。

制御文字

名前	値	説明
NUL	00	ヌル
SOH	01	ヘッダー開始
STX	02	テキスト開始
ETX	03	テキスト終結
EOT	04	伝送終了
ENQ	05	問い合わせ
ACK	06	応答
BEL	07	ベル
BS	08	バックスペース
HT	09	水平タブ
LF	0A	改行
VT	0B	垂直タブ
FF	0C	用紙送り
CR	0D	復帰
SO	0E	シフトアウト
SI	0F	シフトイン
DLE	10	データ・リンク・エスケープ
DC1	11	装置制御 1
DC2	12	装置制御 2
DC3	13	装置制御 3
DC4	14	装置制御 4
NAK	15	否定応答
SYN	16	同期アイドル
ETB	17	伝送ブロック終結
CAN	18	取り消し
EM	1	メディア終了
SUB	1A	置換文字
ESC	1B	エスケープ文字
IS4	1C	情報分離文字 4
IS3	1D	情報分離文字 3
IS2	1E	情報分離文字 2
IS1	1F	情報分離文字 1

図形文字

各コード・セットは、そのそれぞれに固有のコード値が含まれる 1 つ以上の文字セットに分けられます。ISO 規格では、文字のエンコードのために 6 つの列が予約済みであり、制御文字の列で図形文字をエンコードすることはできません。

単一バイトとマルチバイトのコード・セット

1 バイトの 8 ビットすべてを使用するコード・セットでは、ヨーロッパ語、中東語、および他のアルファベット言語をサポートできます。 そのようなコード・セットを、単一バイト・コード・セットと言います。 単一バイト・コード・セットには、制御文字を含まない 191 文字のエンコード制限があります。

191 文字を超える言語では、単一バイト文字 (8 ビット) とマルチバイト文字 (8 ビット超) を組み合わせて使用します。 システムは、1 つの文字をエンコードするために任意のビット数をサポートしています。

ISO コード・セット

次のトピックにリストされたコード・セットは、国際標準化機構 (ISO) によって設定された定義に基づいています。

ISO646-IRV

以下の **ISO646-IRV** コード・セットは、7 ビット・エンコード方式に基づく情報処理に使用されるコード・セットを定義しています。 このコード・セットに関連付けられている文字セットは、 ASCII 文字から派生したものです。

16 進数の 2 番目の桁 ＼ 16 進数の 1 番目の桁	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	BLANK (SPACE)	0	@	P	‘	p								
1	SOH	DC1	!	1	A	Q	a	q								
2	STX	DC2	º	2	B	R	b	r								
3	ETX	DC3	#	3	C	S	c	s								
4	EOT	DC4	$	4	D	T	d	t								
5	ENQ	NAK	%	5	E	U	e	u								
6	ACK	SYN	&	6	F	V	f	v								
7	BEL	ETB	'	7	G	W	g	w								
8	BS	CAN	(	8	H	X	h	x								
9	HT	EM	)	9	I	Y	i	y								
A	LF	SUB	*	:	J	Z	j	z								
B	VT	ESC	+	;	K	[	k	{								
C	FF	IS4	,	<	L	\	l	|								
D	CR	IS3	–	=	M	]	m	}								
E	SO	IS2	.	>	N	^	n	~								
F	S1	IS1	/	?	O	_	o	△								

ISO8859 ファミリー

ISO8859 は、単一バイト・エンコードをベースにしたファミリーで、他の ISO、米国規格協会 (ANSI)、および欧州電子計算機工業会 (ECMA) のコード拡張技法と互換性があります。 ISO8859 エンコードでは、

コード・セットのファミリーが定義されていて、その各メンバーは独自の固有文字セットを含んでいます。 7 ビット ASCII コード・セットは、ISO8859 ファミリーの各コード・セットの適切なサブセットです。

ASCII コード・セットでは英語アルファベットの配列を定義するのに対して、グラフィック右 (GR) 文字は、特定の言語に基づいて配列されていません。 ロケールで、言語固有の配列を定義します。

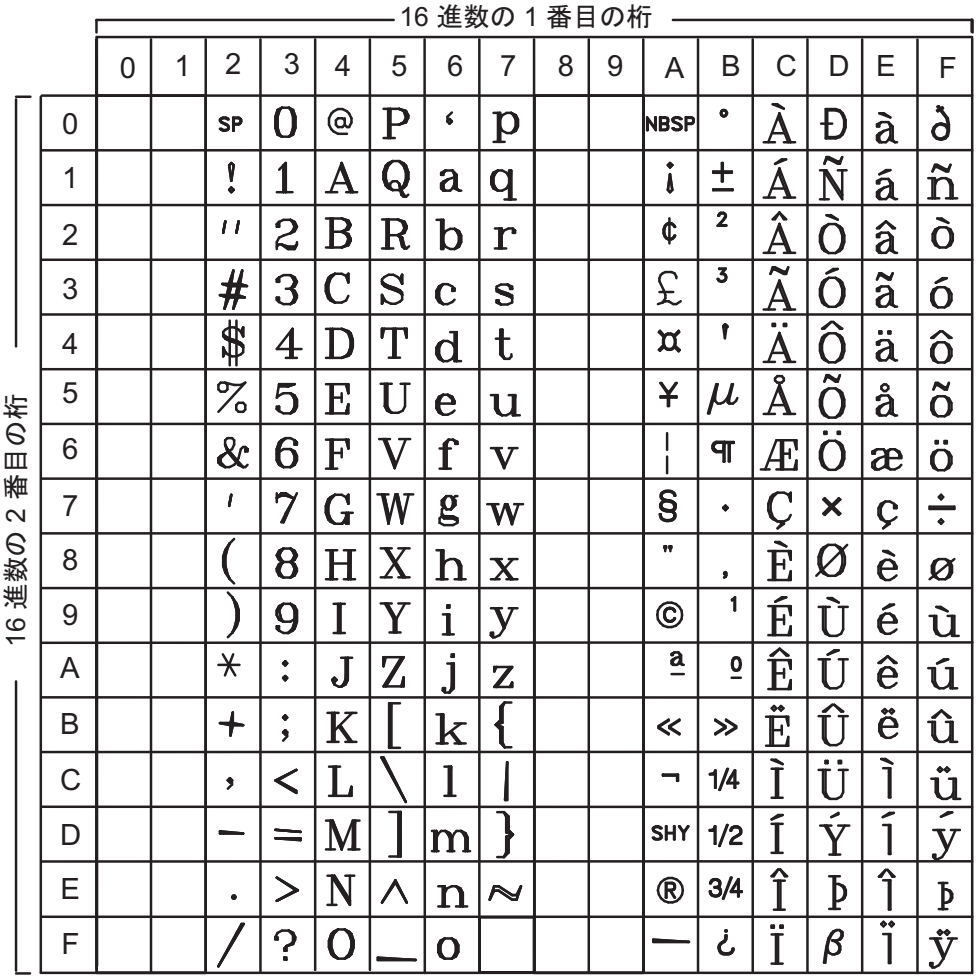
各コード・セットには、ASCII 文字セットと、独自の固有文字セットが含まれています。 ISO8859 エンコードの図には、ISO8859 汎用エンコード・スキームが示されています。

文字エンコード	コード・ポイント	説明	カウント
000xxxxx	00–1F	制御	32
00100000	20	スペース	1
0xxxxxxx	21–7E	7 ビット	94
01111111	7F	削除	1
100xxxxx	80–9F	制御	32
10100000	A0	改行されないスペース	1
1xxxxxxx	A1–F	8 ビット	96

コード・セット ISO8859-1

このセクションでは、コード・セット ISO8859-1 の、使用可能な記号とレイアウトについて記載します。

次の図では、使用可能な記号を要約し、コード・セット ISO8859-1 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 201 ページの『ISO8859–1』 を参照してください。

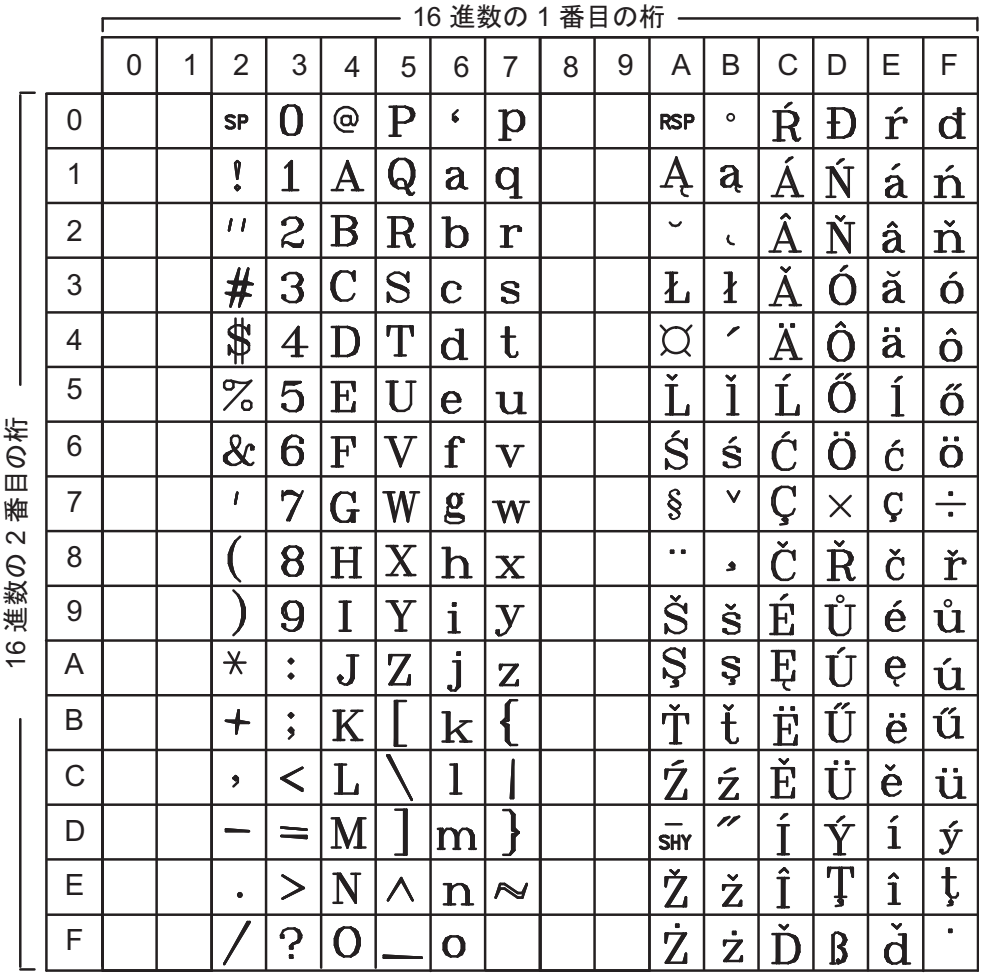


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			NBSP	°	À	Đ	à	ð
1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4			$	4	D	T	d	t			¤	'	Ä	Ô	ä	ô
5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
8			(	8	H	X	h	x			"	¸	È	Ø	è	ø
9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
C			,	<	L	\	l	|			¬	1/4	Ì	Ü	ì	ü
D			-	=	M	]	m	}			SHY	1/2	Í	Ý	í	ý
E			.	>	N	^	n	~			®	3/4	Î	Þ	î	þ
F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

コード・セット ISO8859–2

このセクションでは、コード・セット ISO8859-2 の、使用可能な記号とレイアウトについて記載します。

次の図では、使用可能な記号を要約し、コード・セット ISO8859-2 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 204 ページの『ISO8859–2』 を参照してください。

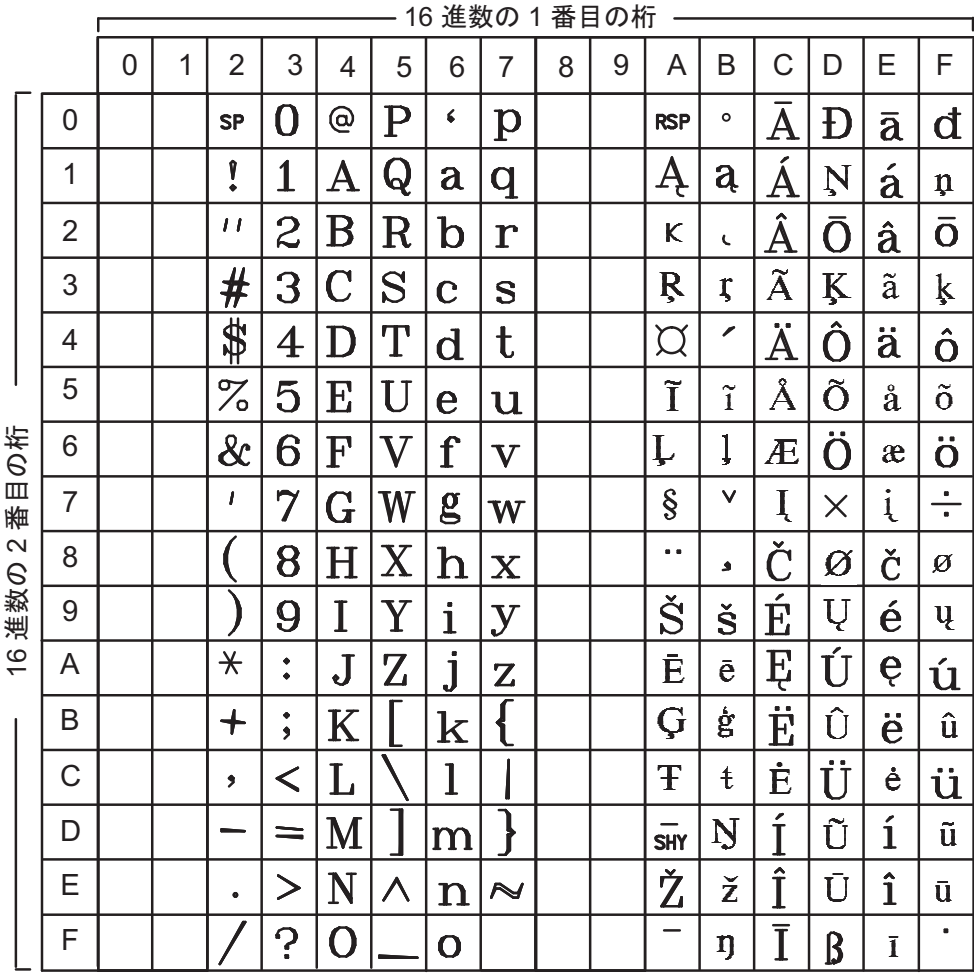


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			RSP	°	Ŕ	Đ	ŕ	đ
1			!	1	A	Q	a	q			Ą	ą	Á	Ń	á	ń
2			"	2	B	R	b	r			˘	˛	Â	Ň	â	ň
3			#	3	C	S	c	s			Ł	ł	Ă	Ó	ă	ó
4			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5			%	5	E	U	e	u			Ľ	ľ	Ĺ	Ő	Í	ő
6			&	6	F	V	f	v			Ś	ś	Ć	Ö	ć	ö
7			'	7	G	W	g	w			§	ˇ	Ç	×	ç	÷
8			(	8	H	X	h	x			¨	¸	Č	Ř	č	ř
9			)	9	I	Y	i	y			Š	š	É	Ů	é	ů
A			*	:	J	Z	j	z			Ş	ş	Ę	Ú	ę	ú
B			+	;	K	[	k	{			Ť	ť	Ë	Ű	ë	ű
C			,	<	L	\	l	|		Ź	ź	Ě	Ü	ě	ü
D			-	=	M	]	m	}			SHY	˝	Í	Ý	í	ý
E			.	>	N	^	n	~			Ž	ž	Î	Ţ	î	ţ
F			/	?	O	_	o				Ż	ż	Ď	ß	ď	˙

コード・セット ISO8859-4

このセクションでは、コード・セット ISO8859-4 の、使用可能な記号とレイアウトについて記載します。

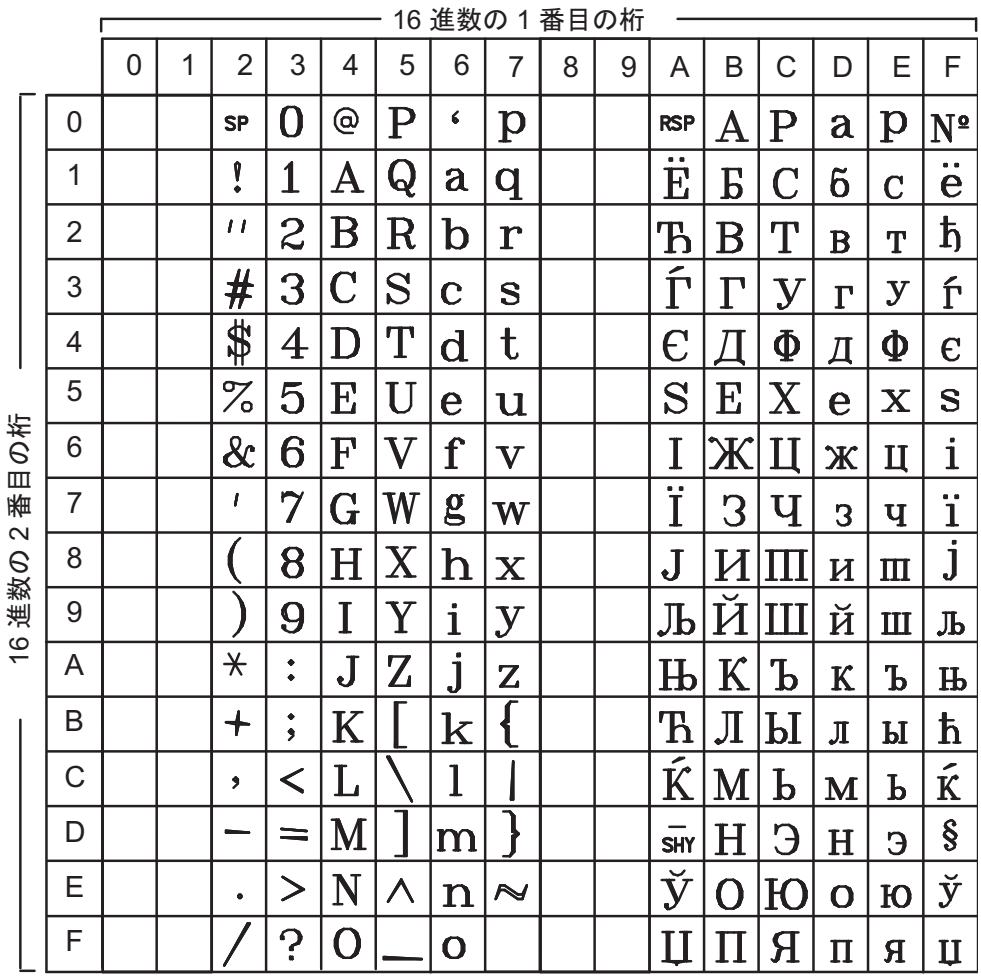
次の図では、使用可能な記号を要約し、コード・セット ISO8859-4 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 208 ページの『ISO8859–4』 を参照してください。



コード・セット ISO8859-5

このセクションでは、コード・セット ISO8859-5 の、使用可能な記号とレイアウトについて記載します。

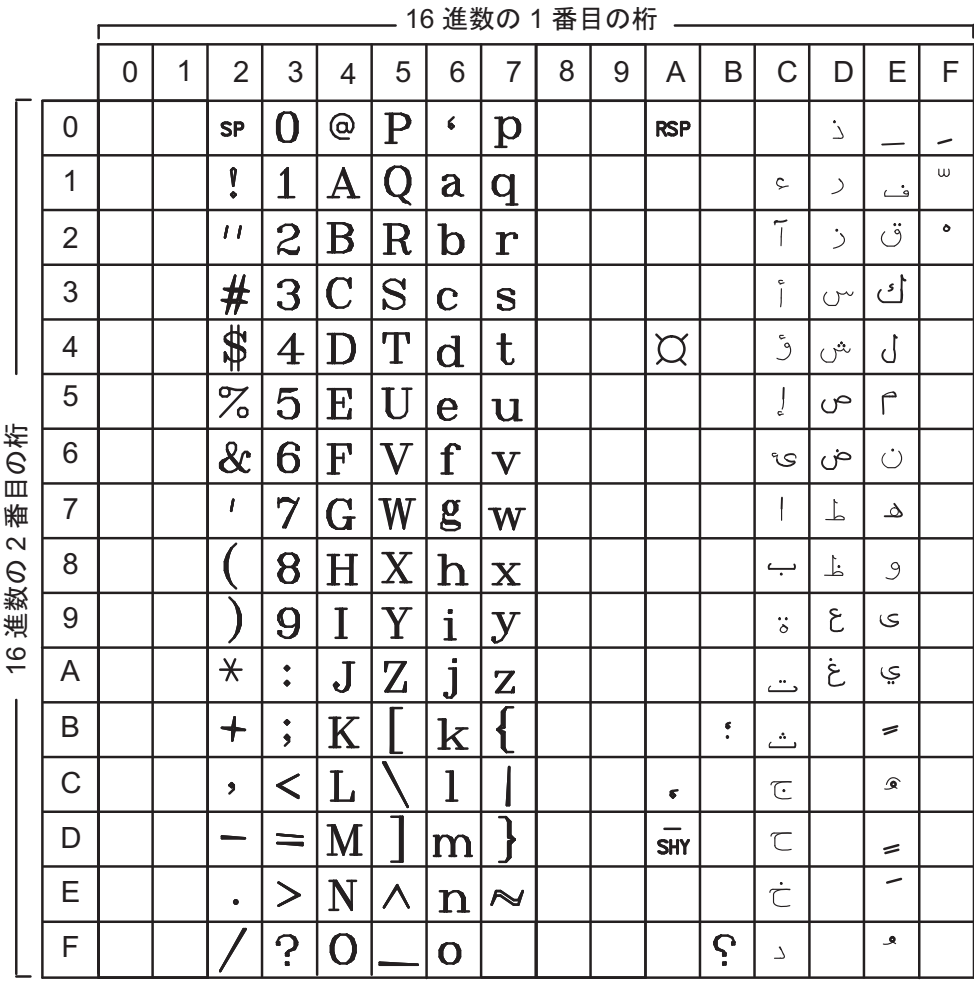
次の図では、使用可能な記号を要約し、コード・セット ISO8859-5 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 210 ページの『ISO8859–5』 を参照してください。



コード・セット ISO8859–6

このセクションでは、コード・セット ISO8859-6 の、使用可能な記号とレイアウトについて記載します。

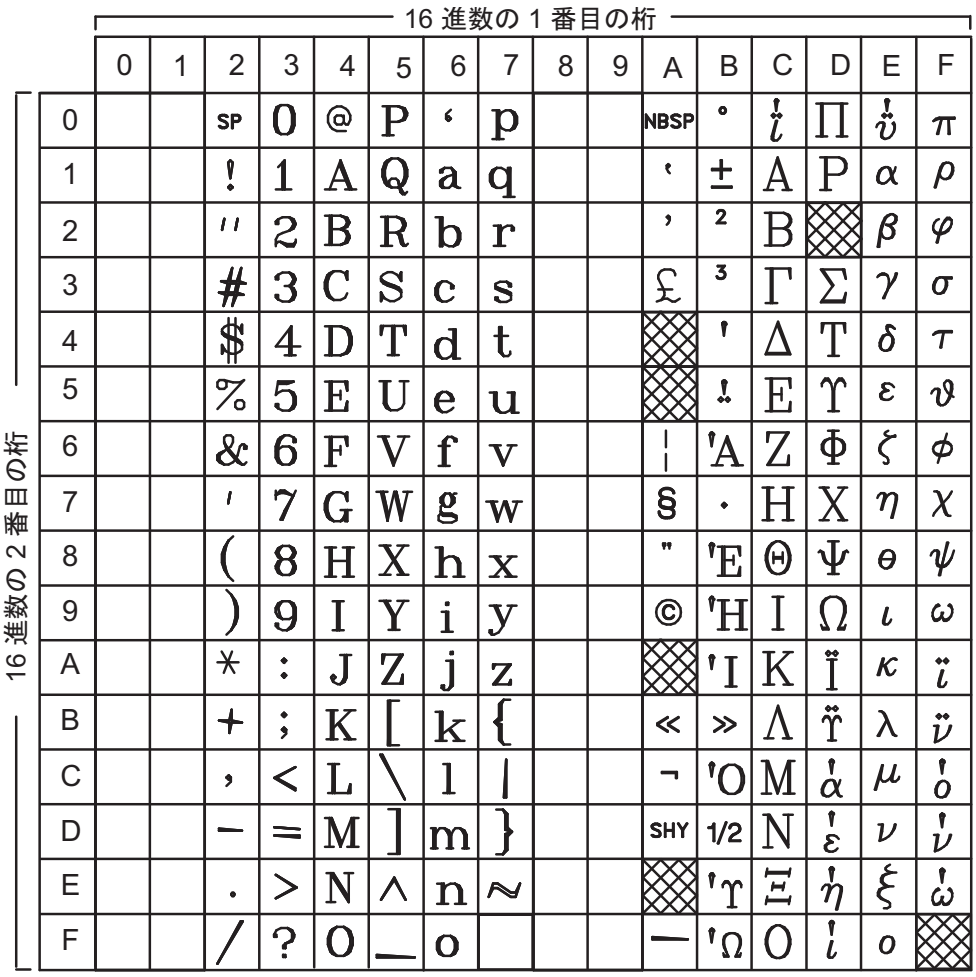
次の図では、使用可能な記号を要約し、コード・セット ISO8859-6 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 213 ページの『ISO8859–6』 を参照してください。



コード・セット ISO8859–7

このセクションでは、コード・セット ISO8859-7 の、使用可能な記号とレイアウトについて記載します。

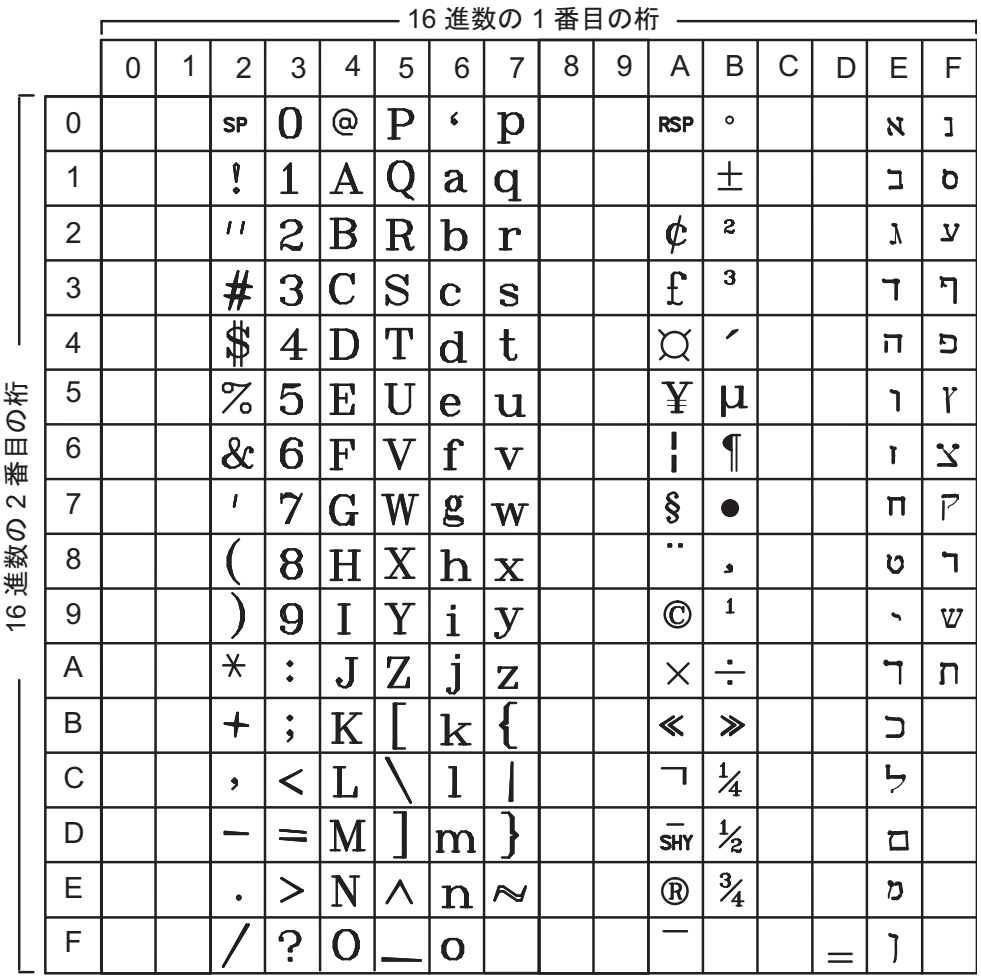
次の図では、使用可能な記号を要約し、コード・セット ISO8859-7 のレイアウトが示されています。 このコード・セットは、ASCII 文字セットと、その独自の固有文字セットで構成されています。 このコード・セットに関するテキストでの説明は、 214 ページの『ISO8859–7』 を参照してください。



コード・セット ISO8859–8

このセクションでは、コード・セット ISO8859-8 の、使用可能な記号とレイアウトについて記載します。

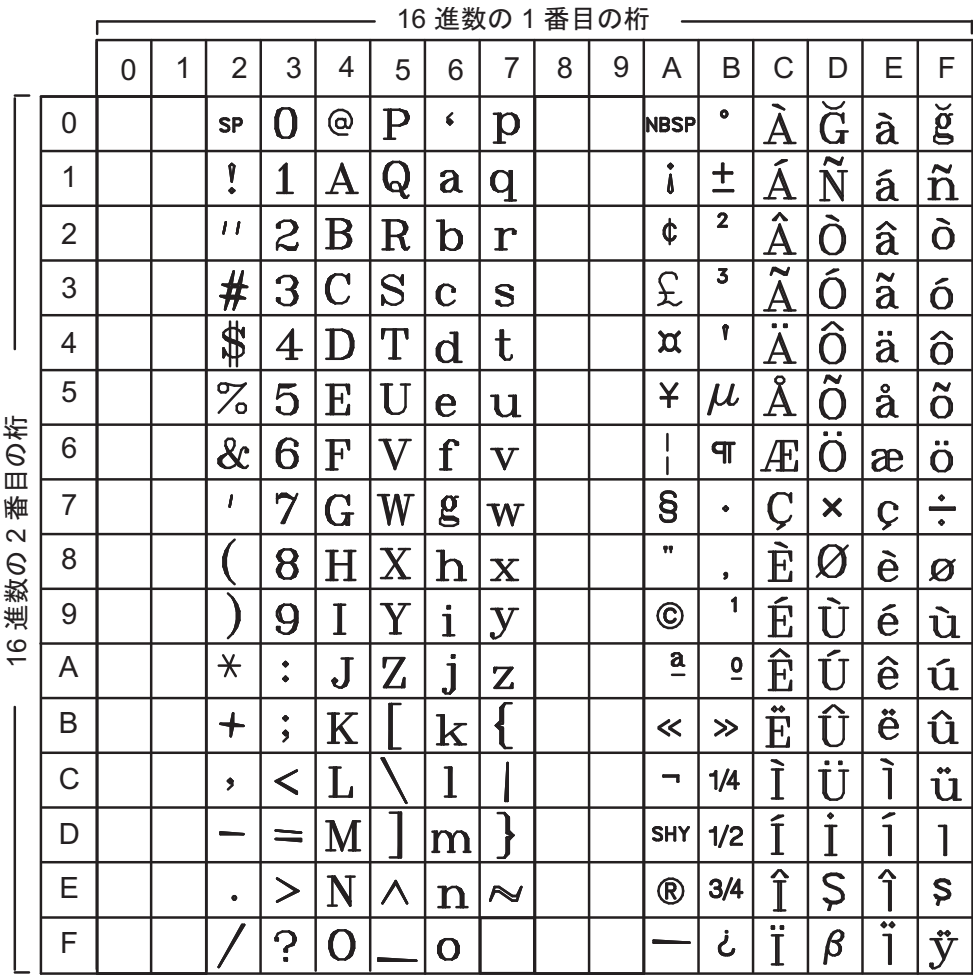
次の図では、使用可能な記号を要約し、コード・セット ISO8859-8 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 217 ページの『ISO8859–8』 を参照してください。



コード・セット ISO8859–9

このセクションでは、コード・セット ISO8859–9 の、使用可能な記号とレイアウトについて記載します。

次の図には、コード・セット ISO8859-9 で使用可能な記号が要約され、コード・セット ISO8859-9 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 219 ページの『ISO8859–9』 を参照してください。

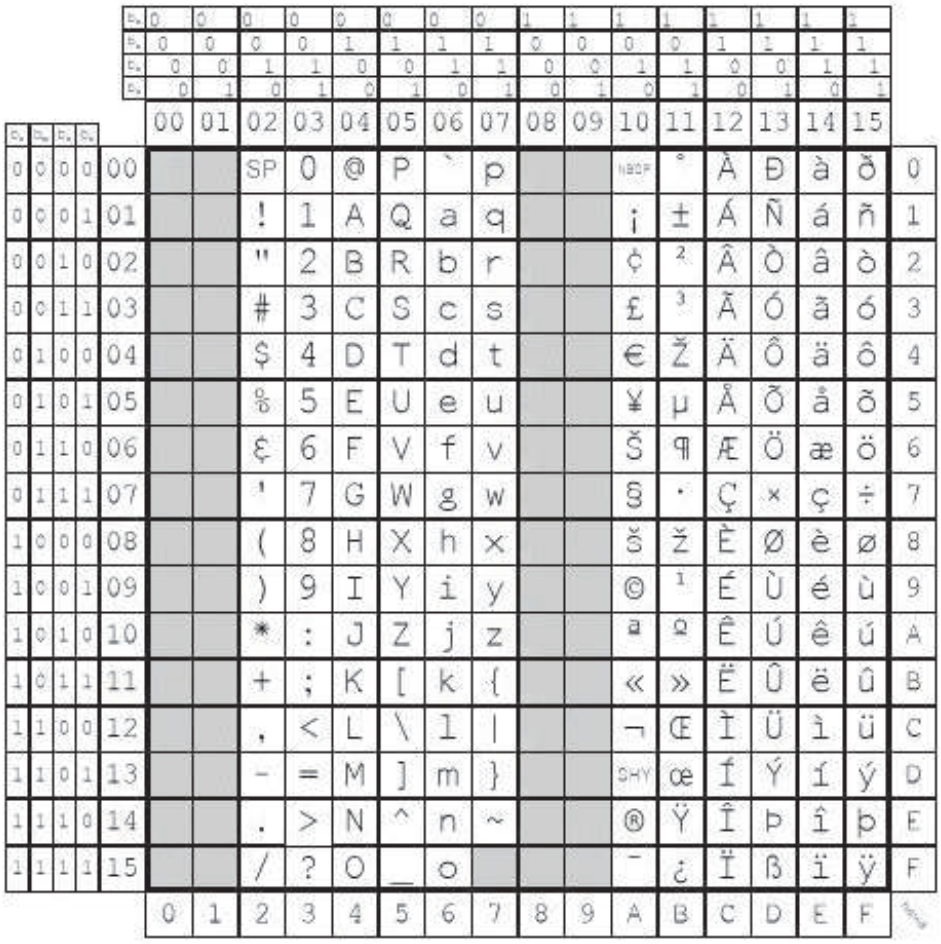


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			NBSP	°	À	Ğ	à	ğ
1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4			$	4	D	T	d	t			¤	'	Ä	Ô	ä	ô
5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
8			(	8	H	X	h	x			"	,	È	Ø	è	ø
9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
C			,	<	L	\	l	|			¬	1/4	Ì	Ü	ì	ü
D			-	=	M	]	m	}			SHY	1/2	Í	İ	í	ı
E			.	>	N	^	n	~			®	3/4	Î	Ş	î	ş
F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

コード・セット ISO8859–15

このセクションでは、コード・セット ISO8859-15 の、使用可能な記号とレイアウトについて記載します。

次の図では、使用可能な記号を要約し、コード・セット ISO8859-15 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 222 ページの『ISO8859–15』 を参照してください。



拡張 UNIX コード (EUC) エンコード・スキーム

EUC エンコード・スキームでは、 1 から 4 つの文字セットをサポートできるエンコード規則群を定義しています。 エンコード規則は、7 ビットおよび 8 ビット・データをエンコードするための、 ISO2022 定義に基づいています。 EUC エンコード・スキームでは、いくつかの文字セットを識別するために制御文字を使用します。 次の表は、すべての EUC エンコードの基本構造を示しています。

EUC	文字エンコード
CS0	0xxxxxxx
CS1	1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx ...
CS2	10001110 1xxxxxxx 10001110 1xxxxxxx 1xxxxxxx 10001110 1xxxxxxx 1xxxxxxx 1xxxxxxx ...

EUC	文字エンコード
CS3	10001111 1xxxxxxx 10001111 1xxxxxxx 1xxxxxxxx 10001111 1xxxxxxx 1xxxxxxxx 1xxxxxxxx ...

EUC という語は、これらの汎用エンコード規則を示しています。 EUC に基づくコード・セットは、EUC エンコード規則に準拠していますが、さらに、特定のインスタンスに関連付けられた特定の文字セットを識別します。例えば、日本語の IBM-eucJP は、EUC エンコード規則に従って、日本工業規格文字のエンコードを参照します。

最初のセット (CS0) には、必ず ISO646 文字セットが含まれています。 他のすべてのセットでは、最大重みビット (MSB) を 1 にセットしなければならず、文字をエンコードするときに任意のバイト数を使用できます。 さらに、セット内のすべての文字に関して、次の条件を満たす必要があります。

- すべての文字をエンコードするために同じバイト数でなければならない
- 同じ列表示幅 (固定幅の端末装置での列数) でなければならない

3 番目のセット (CS2) の文字にはすべて、先頭に制御文字 SS2 (単一シフト 2、 0x8e) が付けられます。EUC に準拠するコード・セットでは、3 番目のセットを識別する目的以外に、 SS2 制御文字を使用することはありません。

4 番目のセット (CS3) の文字にはすべて、先頭に制御文字 SS3 (単一シフト 3 0x8f) が付けられます。EUC に準拠するコード・セットでは、4 番目のセットを識別する目的以外に、 SS3 制御文字を使用することはありません。

IBM-eucJP

日本語の EUC は、単一バイト文字とマルチバイト文字で構成されるエンコードです。 このエンコードは、ISO2022、日本工業規格 (JIS)、および EUC 定義に基づいています。

IBM-eucJP コード・セットは、次の文字セットで構成されています。

文字セット	説明
JISCII	JISX0201 グラフィック左文字セット
JISX0201.1976	カタカナ/ひらがなグラフィック右文字セット
JISX0208.1983	漢字レベル 1 および 2 文字セット
IBM-udcJP	IBM ユーザー定義可能文字

IBM-eucJP コード・セットは、次もサポートできます。

文字セット	説明
JISX0212.1990	補足分の漢字

IBM-eucJP コード・セットは、次のようにエンコードされます。

- CS0 が、0x00 の位置で開始する JISX0201 グラフィック左文字をマップします。
- CS1 が、0xa1xa1 の位置で開始する JISX0208 文字セットをマップします。 CS1 での位置 0xf5a1 から 0xfefe (940 文字) は、 1 次ユーザー定義可能文字域として予約されています。
- CS2 が、0x8ea1 の位置で開始する JISX0201 グラフィック右をマップします。

- CS3 は、0x8fa1a1 の位置で開始する JISX0212 をマップできます。 CS3 での位置 0x8ff5a1 から 0x8ffefe (940 文字) は、 2 次ユーザー定義可能文字域として予約されています。 CS3 での位置 0x8feea1 から 0x8ff4fe (658 文字) は、将来のシステム使用のために予約されています。 したがって、この領域を使用しないようにします。

IBM-eucCN

中国語 (簡体字) 言語の EUC は、1 または 2 バイトを含む文字で構成されたエンコードです。 EUC エンコードは、ISO2022、中華人民共和国で定義された GB2312、およびメーカーに固有のマルチバイト文字定義に基づいています。

現在の GB2312 では、6,763 の中国語 (簡体字) 文字と 682 の記号が定義されています。 IBM-eucCN は、94x94 文字までを含む 1 つの平面という概念に基づいています。 これらの文字のエンコード値は、0xa1a1 から 0xfefe の範囲です。

GB2312 は、EUC の CS1 にマップされます。 特に、IBM-eucCN は、次の文字セットで構成されています。

文字セット	説明
ISO0646-IRV	7 ビット ASCII 文字セット、グラフィック左。
GB2312.1980	7445 文字を含みます。 位置 0xa1a1 から 0xfedf (ユーザー定義文字によっては 0xa1a1 から 0xfedf) を使用します。

IBM-udcCN	GB に散らばっています。 位置 Oxa1a1 から Oxfedf を使用します。 実際の値は、以下のとおりです。
	`a2a1 -- a2b0  a1e3 -- a2e4  a1ef -- a2f0`
	`a2fd -- a1fe  a4f4 -- a4fe  a5f7 -- a5fe`
	`a6b9 -- a6c0  a6d9 -- a6fe  a7c2 -- a7d0`
	`a7f2 -- a7fe  a8bb -- a8c4  a8ea -- a9a3`
	`a9f0 -- affe  a7fa -- d7fe  f8a1 -- fedf`

IBM-sbdCN	GB に散らばっています。 位置 0xfee0 から 0xfefe を使用します。

GB18030

GBK は、Guo (国) Biao (標準) Kuo (拡張) を意味します。 GB18030 は、国定の「業界 GB」定義を拡張したもので、Unicode で定義されたすべての漢字 20,902 文字と、Big-5 コード (中国語 (繁体字) PC のデファクト・スタンダード) で定義されたその他の DBCS 記号を含みます。 GB18030 では、中華人民共和国および台湾で使用されているすべての DBCS 文字と記号が定義されています。

ロケール	コード・セット	説明
Zh_CN	GB18030	中国語 (簡体字)、GB18030 ロケール

コード範囲	語数	マーク
A1A1-A9FE	846	GB2312、GB12345 (GBK/1)
A840-A9A0	192	Big5、記号 (GBK/5)
B0A1-F7FE	6768	GB2312 (GBK/2)
8140-A0FE	6080	GB13000 (GBK/3)
AA40-FEA0	8160	GB13000 (GBK/4)
AAA1-AFFE	564	ユーザー定義 1
F8A1-FEFE	658	ユーザー定義 2
A140-A7A0	672	ユーザー定義 3

IBM-eucTW

中国語 (繁体字) 言語の EUC は、1、2、および 4 バイトを含む文字で構成されたエンコードです。 EUC エンコードは、ISO2022、およびメーカーに固有のマルチバイト文字定義に基づいています。

現在の CNS では、13,501 の漢字と 684 の記号が定義されています。 IBM-eucTW は、15 の平面の概念に基づいていて、それぞれが 8836 (94x94) までの文字を含んでいます。 これらの文字のエンコード値は、0xa1a1 から 0xfefe の範囲です。 現在のところ文字は 4 つの平面だけに定義されていて、他の平面は将来の拡張のために予約されています。

15 の平面が EUC の CS1 および CS2 にマップされていて、そのうち、EUC の CS2 は 14 の平面で構成されています。 特に、IBM-eucTW は、次の文字セットで構成されています。

文字セット	説明
ISO646-IRV	7 ビット ASCII 文字セット、グラフィック左。
CNS11643.1986-1	平面 1、6085 文字 (5401+684) を含む。 この平面では、位置 0ax1a1-0xc2c1 と 0xc4a1-0xfdcb を使用します。
CNS11643.1986-2	平面 2、7650 文字を含む。 この平面では、位置 0x8ea2a1a1-0x8ea2f2c4 を使用します。
CNS11643.1992-3	平面 4、7298 文字を含む。 この平面では、位置 0x8ea4a1a1-0x8ea4eedc を使用します。
IBM-udcTW	平面 12、6204 文字を含む。 この平面は、ユーザー定義文字 (udc) 領域用に予約されています。 位置 0x8eaca1a1-0x8ea2f2c4 を使用します。
IBM-sbdTW	平面 13、325 文字を含む。 この平面は、メーカーに固有な記号のために予約されています。 位置 0xeada1a1-0x8eada4cb を使用します。

平面 3-11 は、位置 0x8ea3xxxx から 0x8eabxxxx を使用することになっています。平面 14-15 は、位置 0x8eaexxxx から 0x8eafxxxx を使用することになっています。

Big5

中国語 (繁体字) big5 ロケール (**Zh_TW**) コード・セットは、 PC の現場で最も広く使用されているコード・セットで、中国語 (繁体字) を使用する国をサポートするのに使用されます。

Big5 コード・セットでは、13056 の文字と 1004 の記号が定義されています。 ここには、CNS11643.192 の 684 個の記号が含まれると同時に、IBM に固有の 325 個の記号も含まれています。

ロケール	コード・セット	説明
Zh_TW	Big5 (IBM-950)	中国語 (繁体字)、Big5 ロケール

Big5 ロケールのコード範囲**:**

Big5 ロケールのコード範囲は、以下の表で定義されます。

プラン	コード範囲	説明
1	A140H - A3E0H	記号および中国語制御コード
1	A440H - C67EH	一般的に使用される文字
2	C940H - F9D5H	あまり一般に使用されない文字
UDF	FA40H - FEFE	ユーザー定義文字
	8E40H - A0FEH	ユーザー定義文字
	8140H - 8DFEH	ユーザー定義文字
	8181H - 8C82H	ユーザー定義文字
	F9D6H - F9F1H	ユーザー定義文字

コード・セット	語数	コード範囲	マーク
一般的に使用される領域	5841	A140-C67E	
あまり一般に使用されない領域	7652	C940-F9D5	
ET 固有領域 (1)	308	C6A1-C878	
ET 固有領域 (2)	7	C8CD-C8D3	
IBM 固有領域	251	F286-F9A0	低いバイト範囲 81-A0
ユーザー定義領域 (1)	785	FA40-FEFE	
ユーザー定義領域 (2)	2983	8E40-A0FE	
ユーザー定義領域 (3)	2041	8140-8DFE	
ユーザー定義領域 (4)	354	8181-8C82	低いバイト範囲 81-AQ
ユーザー定義領域 (5)	41	F9D6-F9FE	

IBM-eucKR

韓国語の EUC は、単一バイト文字とマルチバイト文字で構成されるエンコードです。 このエンコードは、Korean Standard Code Set である ISO2022 と、EUC 定義に基づいています。

韓国語 EUC コード・セットは、以下の主要な文字グループから構成されます。

- ASCII (英語)
- ハングル (韓国語文字)

ハングル・コード・セットには、ハングル文字と漢字 (韓国語) 文字 (元は中国語) が組み込まれています。ハングルの 1 文字は、複数の子音と母音で構成されることがあります。 しかし、ほとんどのハングルの単語は、漢字 (韓国語) で表現できます。 それぞれの漢字 (韓国語) 文字には、独自の意味があり、ハングルよりも特化されています。

IBM-eucKR コード・セットは、次の文字セットで構成されています。

文字セット	説明
ISO646-IRV	7 ビット ASCII 文字セット、グラフィック左
KSC5601.1987-0	韓国語図形文字セット、グラフィック右

IBM PC コード・セット

IBM PC コード・セットは、IBM PC システムと AIX で元々サポートされているコード・セットです。 IBM PC コード・セットは、図形文字を Control One (C1) 制御域に割り当てます。 これらの制御文字に依存するアプリケーションでは、これらのコード・セットをサポートできません。

ASCII 文字は、位置 0x20-0x7e の最大重みビット (MSB) ゼロでエンコードされます。 IBM PC 固有文字セットと結び付けられた拡張 Latin 1 は、位置 0x80-0xfe でエンコードされる拡張文字セットを構成します。 次の表は、IBM-850 コード・セットでの制御文字、ASCII 文字、および外字のロケーションを示しています。

文字エンコード	コード・ポイント	説明	カウント
000xxxxx	00–1F	制御	32
00100000	20	スペース	1
0xxxxxxx	21–7E	7 ビット	94
01111111	7F	削除	1
1xxxxxxx	80–FE	8 ビット	17
11111111	FF	すべて 1	1

IBM PC 固有の文字セットには以下のものが含まれています。

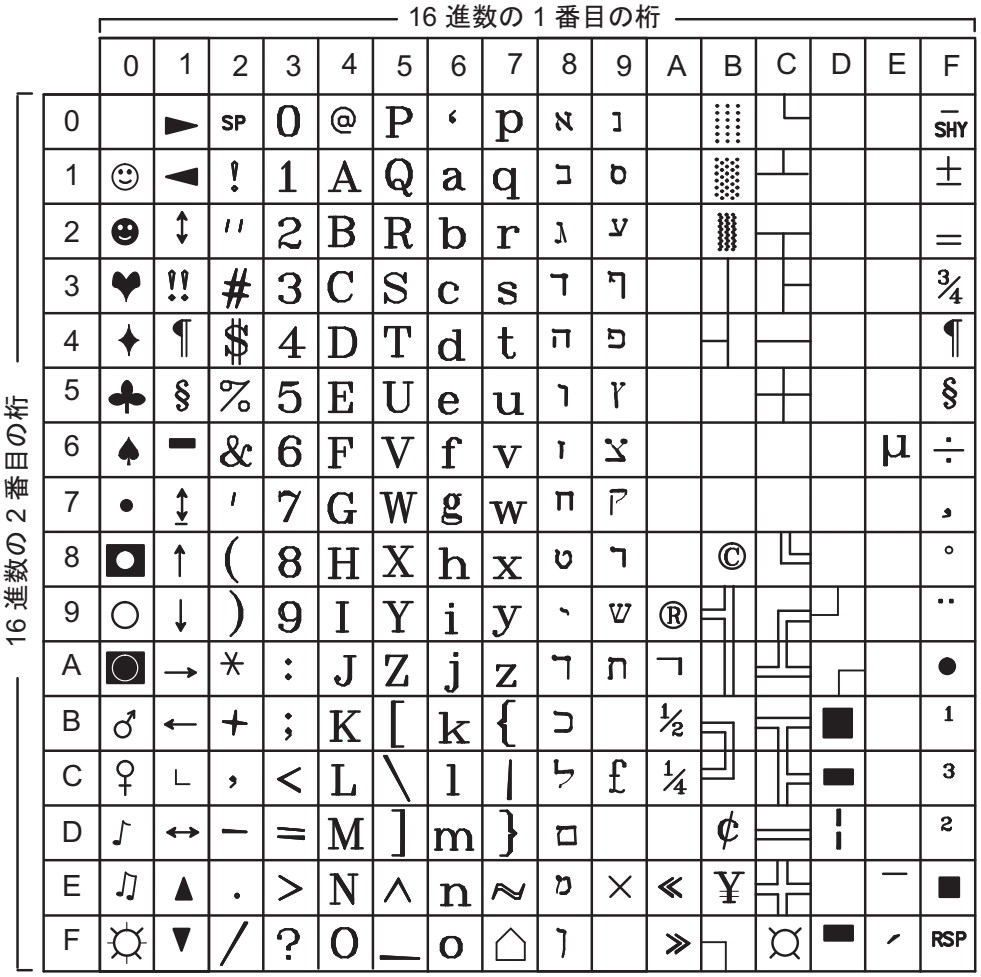
表 *4. IBM PC* 固有の文字セット

シンボル	戻りコード
フロリン記号	0x9f
クオーター・ハッシュ	0xb0
ハーフ・ハッシュ	0xb1
フル・ハッシュ	0xb2
垂直バー	0xb3
右側中間	0xb4
二重右側中間	0xb9
二重垂直バー	0xba
二重右上角ボックス	0xbb
二重右下角ボックス	0xbc
右上角ボックス	0xbf
左下角ボックス	0xc0
下側中間	0xc1
上側中間	0xc2
左側中間	0xc3
中央ボックス・バー	0xc4
交点	0xc5
二重左下角ボックス	0xc8
二重左上角ボックス	0xc9
二重下側中間	0xca
二重上側中間	0xcb
二重左側中間	0xcc
二重中央ボックス・バー	0xcd
二重交点	0xce
スモール i ドットレス	0xd5
右下角ボックス	0xd9
左上角ボックス	0xda
高輝度文字セル	0xdb
高輝度文字セル - 下半分	0xde
高輝度文字セル - 上半分	0xdf
上線	0xee
中間ドット、製品ドット	0xfa
垂直塗りつぶし長方形	0xfe

IBM-856

このセクションでは、コード・セット IBM-856 の、使用可能な記号とレイアウトについて記載します。

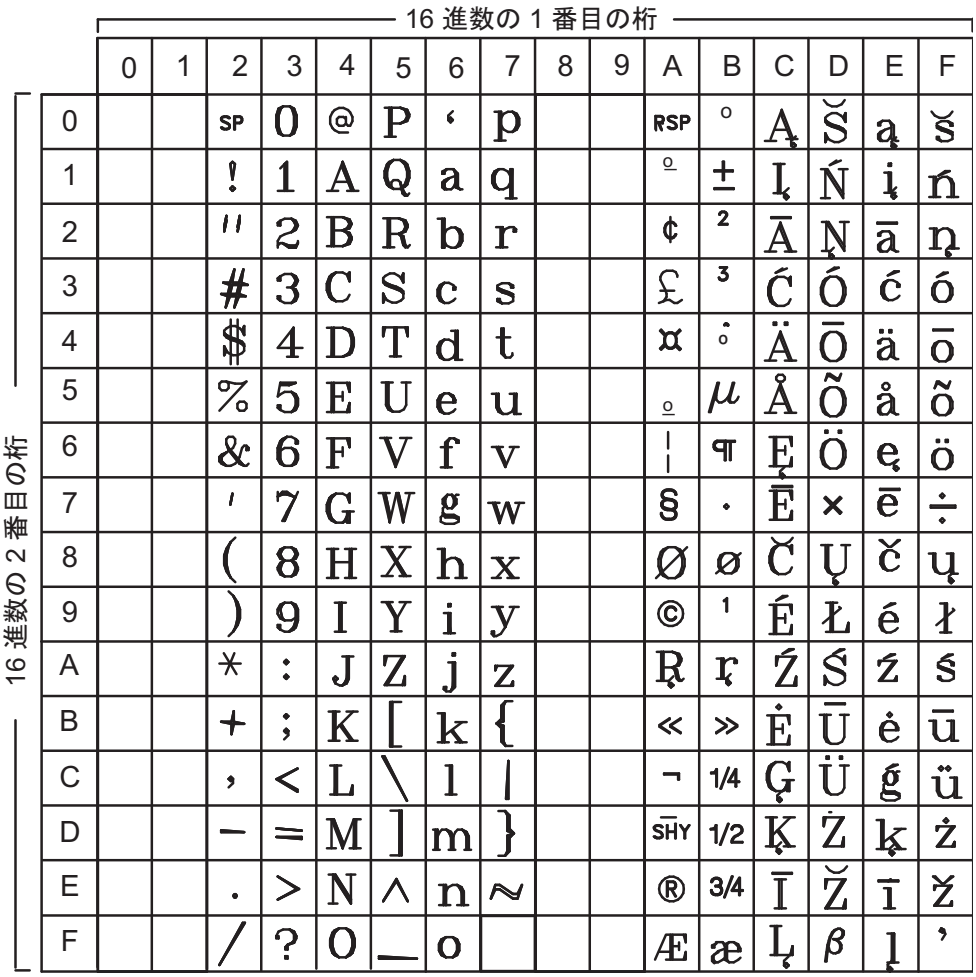
次の図では、使用可能な記号を要約し、コード・セット IBM-856 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 224 ページの『IBM-856』 を参照してください。



IBM-921

このセクションでは、コード・セット IBM-921 の、使用可能な記号とレイアウトについて記載します。

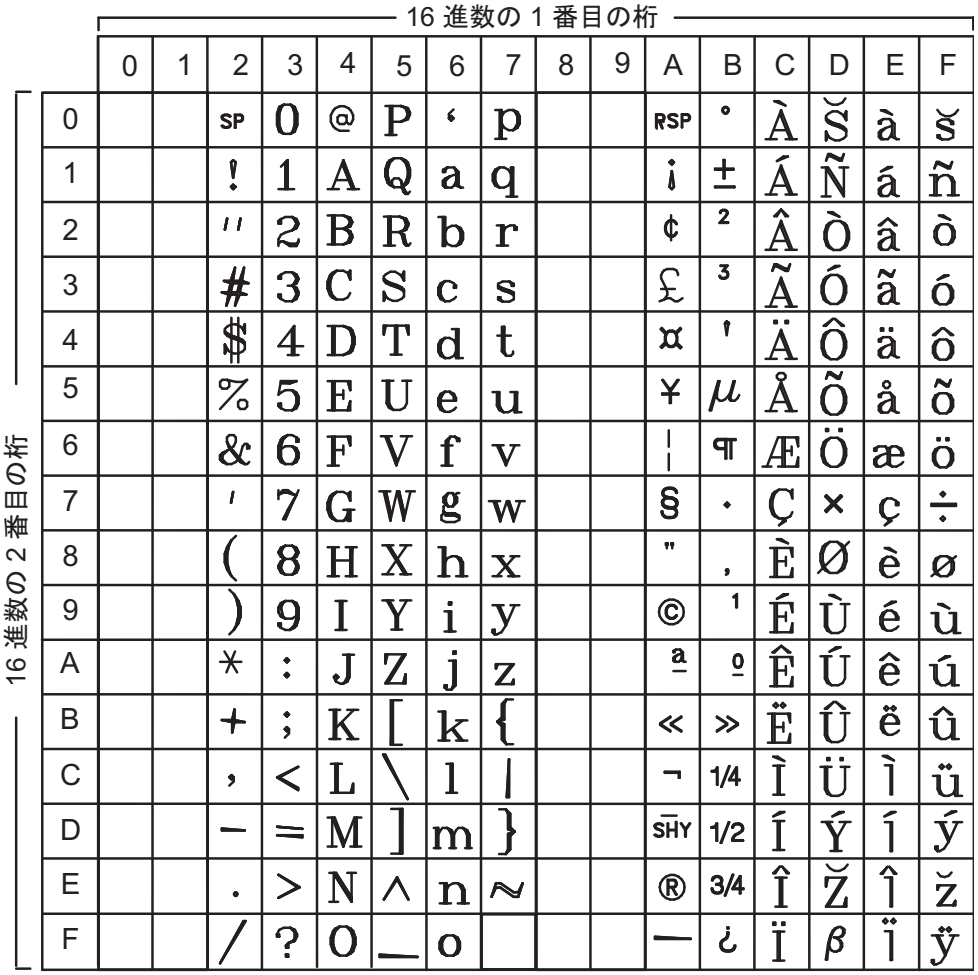
次の図では、使用可能な記号を要約し、コード・セット IBM-921 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 226 ページの『IBM-921』 を参照してください。



IBM-922

このセクションでは、コード・セット IBM-922 の、使用可能な記号とレイアウトについて記載します。

次の図では、使用可能な記号を要約し、コード・セット IBM-922 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 230 ページの『IBM-922』 を参照してください。



IBM-943 および IBM-932

それぞれの日本語 IBM PC コード・セットは、単一バイトおよびマルチバイト・コード化文字で構成されるエンコードです。 このエンコードは、IBM PC コード・セットに基づくものであり、JIS 文字をシフト位置に配置します。 このことは、シフト *JIS* または SJIS と呼ばれます。

IBM-943 は、IBM-932 より新しい日本語ロケールのコード・セットです。 IBM-943 は、日本語 Microsoft Windows 環境の互換コード・セットです。このコード・セットは、1983 配列シフト JIS と呼ばれます。 IBM-932 と IBM-943 の相違点は、次のとおりです。

- 以前の JIS シーケンス (1978 配列) は IBM-932 に対して適用されるのに対し、新しい JIS シーケンス (1983 配列) は IBM-943 に対して適用されます。
- NEC 選択文字が IBM-943 に追加されています。
- NEC の IBM 選択文字が IBM-943 に追加されています。

IBM-932 コード・セットは、次の文字セットで構成されています。

文字セット	説明
JISCII	JISX0201 グラフィック左文字セット
JISX0201.1976	カタカナ/ひらがなグラフィック右文字セット
JISX0208.1983	漢字レベル 1 および 2 文字セット
IBM-udcJP	IBM ユーザー定義可能文字

IBM-943 コード・セットは、次の文字セットで構成されています。

文字セット	説明
JISCII	JISX0201 グラフィック左文字セット
JISX0201.1976	カタカナ/ひらがなグラフィック右文字セット
JISX0208.1990	漢字レベル 1 および 2 文字セット
IBM-udcJP	IBM ユーザー定義可能文字と、NEC の IBM 選択文字および NEC 選択文字

それぞれの文字の最初のバイトは、指定した文字のバイト数を判別するときに使用されます。 値 0x20-0x7e および 0xa1-oxdf は、例外はありますが JISX0201 文字をエンコードするときに使用されます。 位置 0x81-0x9f と 0xe0-0xfc は、マルチバイト文字の最初のバイトとして使用するために予約されています。 JISX0208 文字は、0x8140 で開始するマルチバイト値にマップされます。 マルチバイト文字の 2 番目のバイトは、どのような値にでもなります。 シフト JIS 表には、これらの文字がコード・セットのどこにあるのかが示されています。

文字エンコード	コード・ポイント	説明	カウント
000xxxxx	00–1f	制御	32
00100000	20	スペース	1
0xxxxxxx	21–7E	7 ビット ASCII	94
01111111	7F	削除	1
10000000	80	未定義	1
100xxxxx 01xxxxxx	[81–9F] [40–7E]	2 バイト	1953
100xxxxx 1xxxxxxx	[81–9F] [80–FC]	2 バイト	3975
10100000	A0	未定義	1
1xxxxxxx	A1–DF	7 ビット単一バイト	63
111xxxxx 01xxxxxx	[E0–FC] [40–7E]	2 バイト	1827
111xxxxx 1xxxxxxx	[E0–FC] [80–FC]	2 バイト	3625
11111101	FD	未定義	1
11111110	FE	未定義	1
11111111	FF	未定義	1

次の表には、IBM-943 の DBCS 部分が示されています。

コード・ポイント	説明
[81–84] [40–7E] および [81–84] [80–F0]	JIS X 0208 (非漢字)
[87] [40–7E] および [87] [80–F0]	NEC 選択文字
[89–98] [40–7E] および [88] [9F-F0]、[89–97] [80–F0]、[98] [80–9F]	JIS X0208 (レベル 1 漢字)
[99–9F] [40–7E] および [98] [9F-F0]、[99–9F] [80–F0]	JIS X0208 (レベル 2 漢字)
[E0–EA] [40–7E] および [E0–EA] [80–F0]	JIS X0208 (レベル 2 漢字)
[ED–EE] [40–7E] および [ED–EE] [80–F0]	NEC IBM 選択文字
[F0–F9] [40–7E] および [F0–F9] [80–F0]	ユーザー定義文字

コード・ポイント	説明
[FA] [40–5C]	IBM 選択文字 (非漢字)
[FA] [5C-7E]、[FB-FC] [40–7E] および [FA-FC] [80–F0]	IBM 選択文字 (漢字)

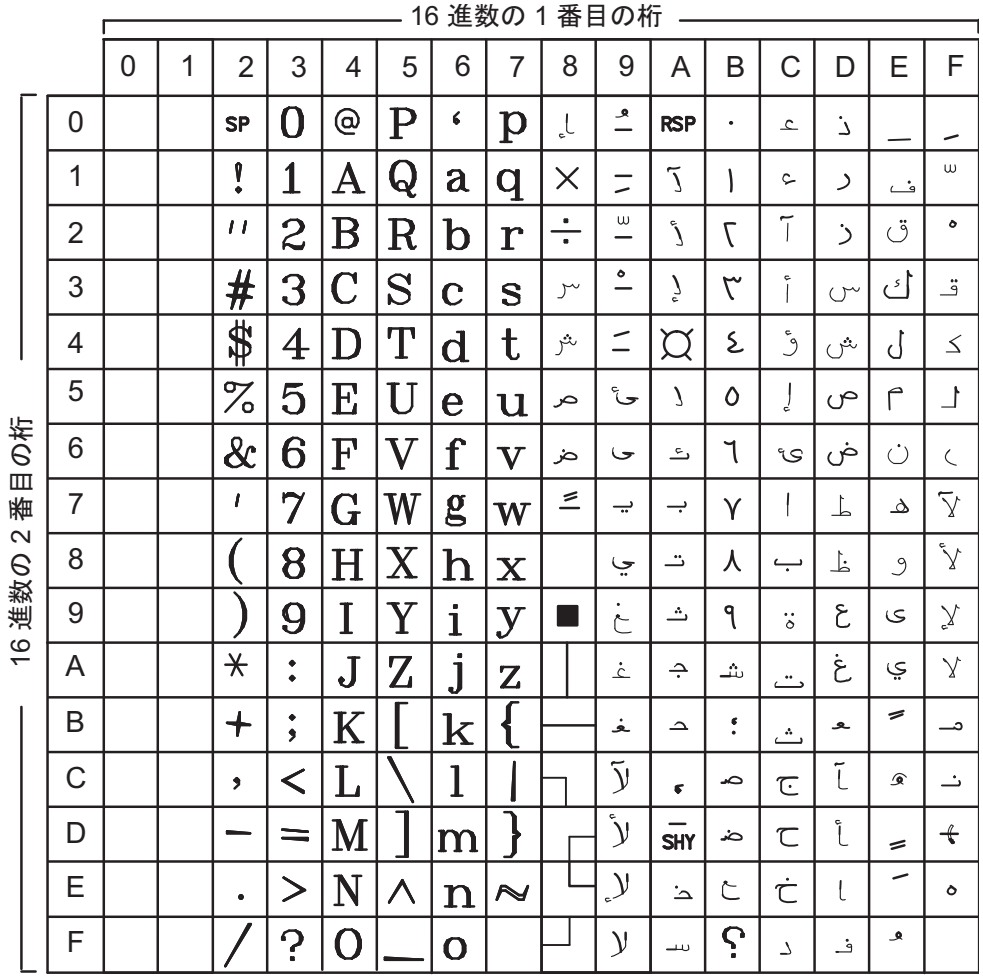
次の表には、IBM-932 の DBCS 部分が示されています。

コード・ポイント	説明
[81–98] [40–7E] および [81–97] [80–FC]、[98] [80–9F]	JIS X 0208 (レベル 1 漢字)
[99–9F] [40–7E] および [98] [9F-FC]、[99–9F] [80–FC]	JIS X 0208 (レベル 2 漢字)
[E0–EF] [40–7E] および [E0–EF] [80–FC]	JIS X 0208 (レベル 2 漢字)
[F0–F9] [40–7E] および [F0–F9] [80–FC]	ユーザー定義文字
[FA–FC] [40–7E] および [FA–FC] [80–FC]	IBM 選択文字

IBM-1046

このセクションでは、コード・セット IBM-1046 の、使用可能な記号とレイアウトについて記載します。

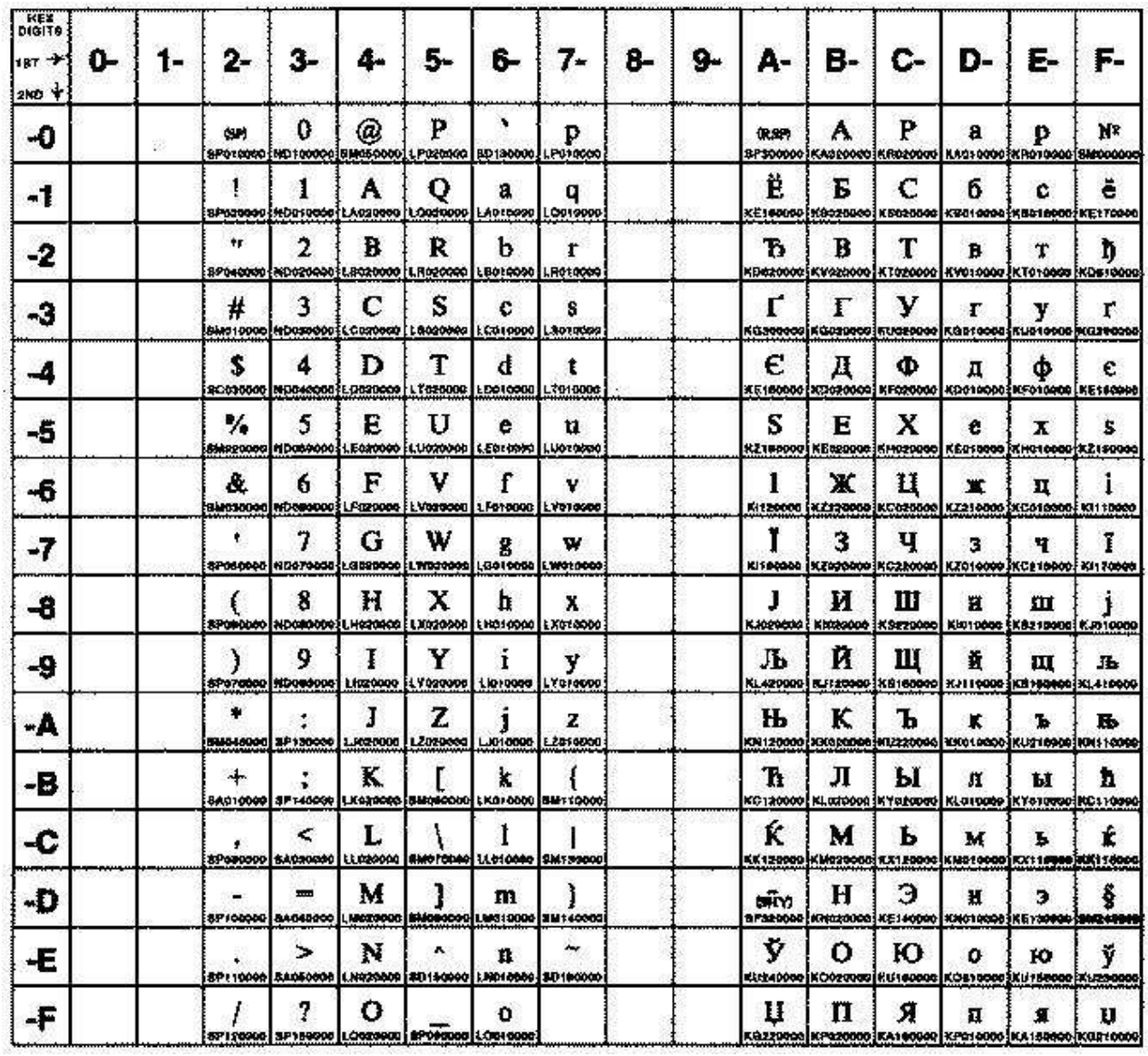
次の図では、使用可能な記号を要約し、コード・セット IBM-1046 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 232 ページの『IBM-1046』 を参照してください。



IBM-1124

このセクションでは、コード・セット IBM-1124 の、使用可能な記号とレイアウトについて記載します。

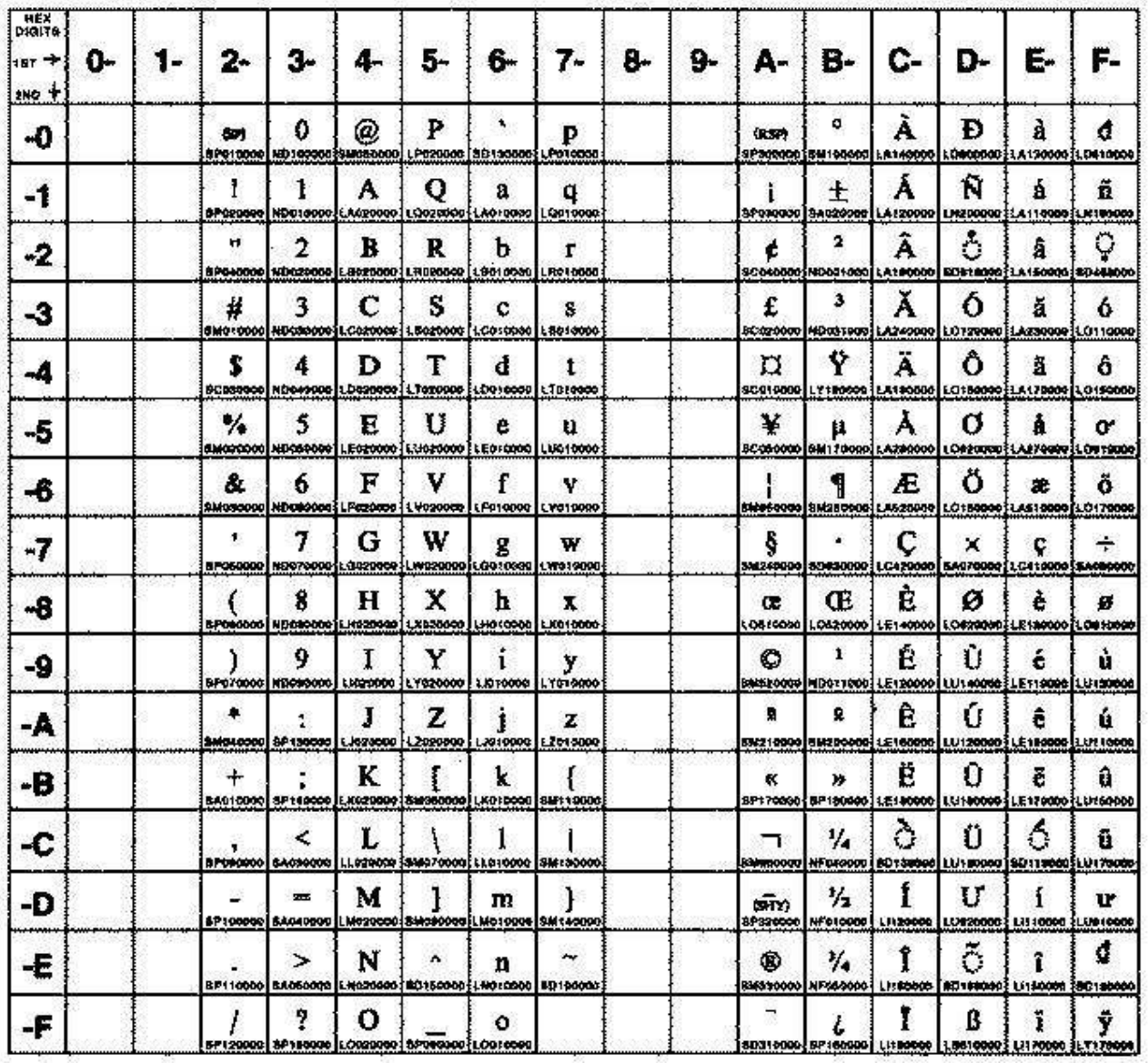
次の図では、使用可能な記号を要約し、コード・セット IBM-1124 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 235 ページの『IBM-1124』 を参照してください。



IBM-1129

このセクションでは、コード・セット IBM-1129 の、使用可能な記号とレイアウトについて記載します。

次の図では、使用可能な記号を要約し、コード・セット IBM-1129 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 238 ページの『IBM-1129』 を参照してください。

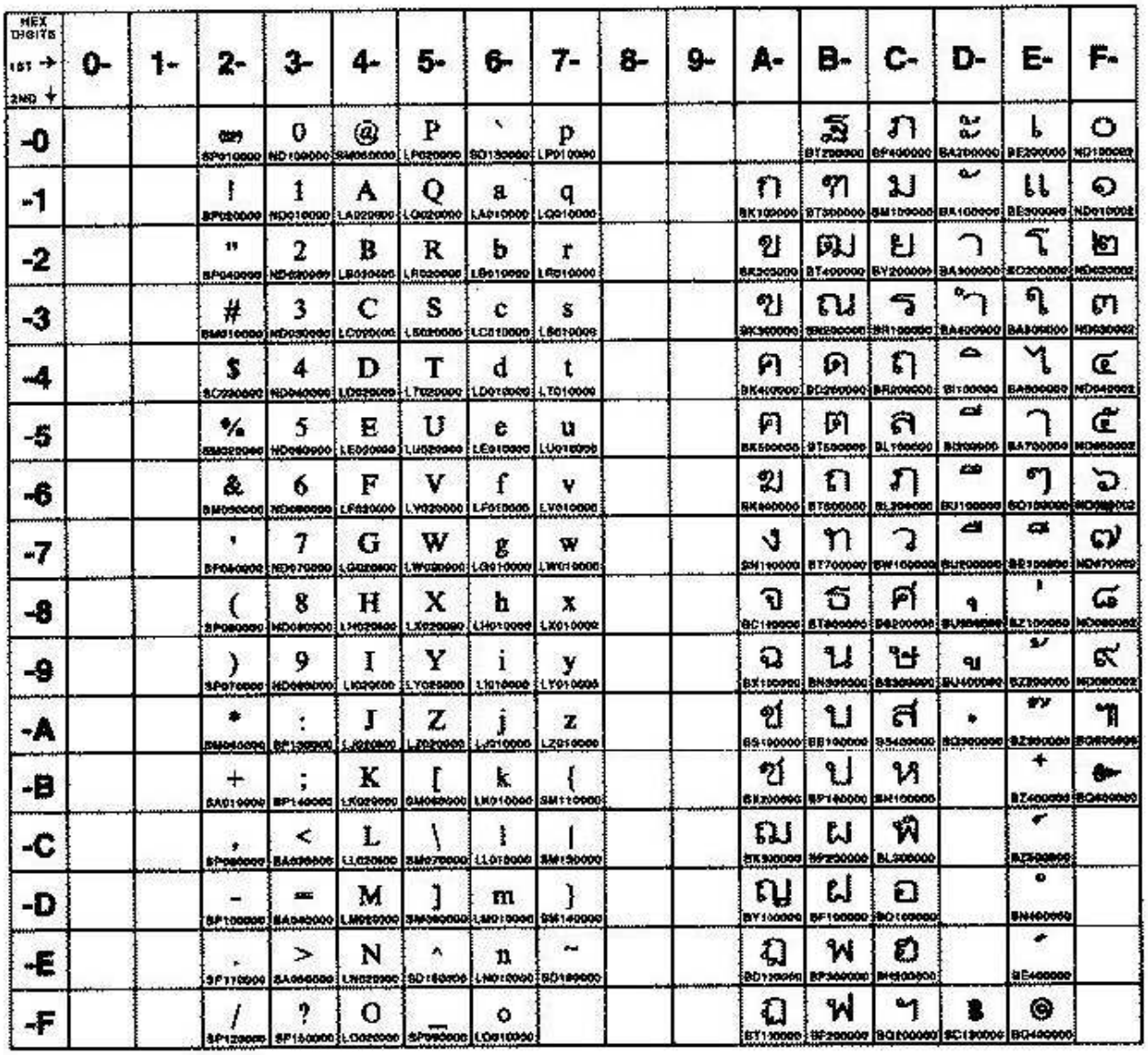


HEX DIGITS 1ST → 2ND ↓	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			(SP)	0	@	P	`	p			(RSP)	°	À	Ð	à	ð
-1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
-2			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
-3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
-4			$	4	D	T	d	t			¤	Ÿ	Ä	Ô	ä	ô
-5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
-6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
-7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
-8			(	8	H	X	h	x			œ	Œ	È	Ø	è	ø
-9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
-A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
-B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
-C			,	<	L	\	l	|		¬	¼	Ð	Ü	Ó	ü
-D			-	=	M	]	m	}			(SHY)	½	Í	Ý	í	ý
-E			.	>	N	^	n	~			®	¾	Î	Õ	î	ð
-F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

TIS-620

このセクションでは、コード・セット TIS-620 の、使用可能な記号とレイアウトについて記載します。

次の図では、使用可能な記号を要約し、コード・セット TIS-620 のレイアウトが示されています。 このコード・セットに関するテキストでの説明は、 241 ページの『TIS-620』 を参照してください。



UCS-2 および UTF-8

AIX には、特定言語または言語グループの必要を満たすコード・セット群が用意されています。 ISO8859 ファミリーのコード・セット、PC コード・セット、または拡張 UNIX コード (EUC) コード・セット内に表現されているコード・セットでは、異なるスクリプトの文字を混合させることはできません。
ISO8859-1 を使用すると、Latin 1 文字 (基本的に、米国、カナダ、西欧、およびラテンアメリカで話されている言語) を混合し表現できます。 ISO8859-2 は西ヨーロッパ言語を、ISO8859-5 はキリル文字を、ISO8859-6 はアラビア語を、 ISO8859-7 はギリシャ語を、ISO8859-8 はヘブライ語を、ISO8859-9 はトルコ語を、 IBM-eucJP は日本語を、IBM-eucKR は韓国語を、IBM-eucTW は中国語 (繁体字) を網羅します。 つまり、上記のコード・セットの中で、すべての言語が網羅されているものはないということです。

国際標準化機構 (ISO) は、Unicode を ISO10646 Universal Multiple-Octet Coded Character Set (UCS-2) の 2 オクテット形式のエンコード方式として採用することにより、コード・セットごとに言語の

網羅が限定されているという問題に取り組みました。 ISO10646 の 32 ビット形式は、4 オクテット形式の UCS-4 として知られています。 AIX では、16 ビット形式の ISO10646 を使用し、標準ラベル *UCS-2* を使用してこのエンコードを記述します。

UCS-2 は、内部処理コードには理想的ですが、AIX などの従来のバイト指向システム上のプレーン・テキストをエンコードするには適しません。 したがって、外部ファイル・コードは、The Open Group の File System Safe UCS Transformation Format (FSS-UTF) になります。 この変換形式によるエンコードは、UTF-8 としても知られています。 *UTF-8* とは、AIX でのこのエンコードに使用されるラベルです。

関連概念:

112 ページの『UTF-8 交換コンバーター』
このセクションでは、各コード・セットと UTF-8 の両方向に行われる変換について説明します。

ISO10646 UCS-2 (Unicode)

Universal Coded Character Set (UCS) は、表現、交換、処理、ストレージ、項目、および世界中の主要言語すべての書き方の表現に対して、1 つのコードを定義する ISO10646 規格の名前です。

UCS-2 の文字コード値は、Unicode Consortium で公開されている Unicode 文字エンコード規格の文字コード値と同じです。 UCS-2 では、すべての主要な書き言語で使用されている文字のコードを定義しています。 UCS-2 では、一連の化学記号、数術記号、および出版記号に加えて、次のスクリプトを網羅しています。

- アラビア語
- アルメニア語
- アゼルバイジャン語
- ベンガル語
- Bopomofo
- キリル文字
- デーバナーガリー文字
- グルジア語
- ギリシャ語
- グジャラート語
- グルムキー文字
- ハングル
- 中国語漢字
- ヘブライ語
- ひらがな
- 国際表音文字 (IPA)
- カタカナ
- 日本語漢字
- カンナダ語
- 韓国語漢字
- ラオ語
- ラテン語
- マラヤーラム文字

- マルタ語
- オリヤー語
- タミール語
- テルグ語
- タイ語
- チベット語
- ウルドゥー語
- ウェールズ語

上記のスクリプトで文字を表示する AIX の機能は、フォントが使用できるかどうかによって制限を受けます。 AIX には、世界のほとんどの主要言語に対するビットマップ・フォントと、Unicode ベースのスケーラブル TrueType フォントが用意されています。

UCS-2 は、多数の組文字をエンコードします。これは、浮動分音記号用の非スペース・マークとも呼ばれます。 これらの文字は、インド語派、タイ語、アラビア語、およびヘブライ語を含む、いくつかのスクリプトで必要です。 組み文字は、ラテン語、キリル文字、およびギリシャ語のスクリプトで文字を生成するのに使用されます。しかし、組み文字が存在すると、同じテキストで別のコーディングが生成される可能性があります。 コーディングがあいまいで、データ保全性が保持されている場合でも、組み文字を含むテキストの処理は非常に複雑です。 組み文字を扱わないよう選択するアプリケーションに適合させるため、ISO10646 は、次のインプリメンテーション・レベルを定義しています。

レベル **1**
: 組み文字を許可しない。

レベル **2**
: タイ語、インド語派、ヘブライ語、およびアラビア語スクリプトからの組のマークを許可します。

レベル **3**
: ラテン語、キリル文字、およびギリシャ語用のマークを含め、組マークを許可します。

注**:** AIX オペレーティング・システムでは、ISO10646-1 ラベルは UCS-2 エンコードを参照します。 このラベルは、UCS-2 の別名として使用できます。

UCS-4 および UTF-32

Unicode 規格は、世界中で一般的に使用されているほとんどの言語のために、標準文字エンコードを定義するのに使用されます。 この規格の 2 バイト形式のことを、一般に *UCS-2* と言います。 しかし、UCS-2 では、2 バイトの数量として、最大で 65,536 文字を表現できるだけです。 Unicode の 4 バイト形式は、*UCS-4* または *UTF-32* と呼ばれ、 Unicode の完全な拡張機能を定義することができ、最大で 1,000,000 を超える固有の文字を定義できます。

UTF-8 (UCS transformation format)

The Open Group は、既存のファイル・システムで使用するよう設計された UCS の変換形式を開発しました。 その意図は、UCS を変換形式の処理コードとして、ファイル・コードとして使用できるようにすることです。

UTF-8 には、以下の属性があります。

- ASCII のスーパーセットです。ここで、ASCII 文字は、同じ数値の単一バイト文字としてエンコードされます。
- マルチバイト文字では、ASCII 文字を表現する値以外には、 ASCII コード値は発生しません。

- 文字の最初のバイトは、マルチバイト文字シーケンスで続くバイト数を示し、シーケンスの別の場所に置くことはできません。

UTF-8 は、1、2、3、4、5、および 6 バイトの長さのマルチバイト文字を使用して、 0 から 0x7FFFFFFF の範囲で UCS 値をエンコードします。単一バイト文字は、0 から 0x7f の範囲の ASCII 文字のために予約されています。 これらのすべての文字では、高い順序のビットが 0 に設定されています。 1 バイトより大きいすべての文字エンコードについては、最初のバイトで、使用されるバイト数が判別され、各バイトの高い順序のビットが設定されます。 10xxxxxx (x はビットを表し、0 か 1) というビットの組み合わせで開始しないバイトはすべて、 UCS 文字シーケンスの先頭になります。 次の表は、UTF-8 マルチバイト・コードを示します。

バイト	ビット	最小 **16** 進数	最大 **16** 進数	バイナリーでのバイト・シーケンス
1	7	00000000	0000007F	0xxxxxxx
2	11	00000080	000007FF	110xxxxx 10xxxxxx
3	16	00000800	0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
4	21	00010000	001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
5	26	00200000	03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxx 10xxxxxx
6	31	04000000	7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

UCS 値は、マルチバイト・エンコードでは、x ビットを連結しただけのものです。 値 (例えば、UCS 0) をエンコードする複数の方法がある場合、最短のエンコードだけが許可されます。

UCS-2 をエンコードするには、次に示す UTF-8 のサブセットが使用されます。

バイト	ビット	最小 **16** 進数	最大 **16** 進数	バイナリーでのバイト・シーケンス
1	7	00000000	0000007F	0xxxxxxx
2	11	00000080	000007FF	110xxxxx 10xxxxxx
3	16	00000800	0000FFFF	1110xxxx 10xxxxxx 10xxxxxx

この UTF-8 のサブセットでは、最大で 3 バイトが必要です。

UTF-16

UTF-16 は、グループ 00 の 16 平面のための、UCS Transformation Format です。 UTF-16 は、代理を使用する Unicode 規格と同等の ISO/IEC エンコードです。 UTF-16 では、それぞれの UCS-2 コード値がそのものを表します。 ISO/EIC 10646 の平面 1..16 における非 BMP コード値は、特別なコードの組み合わせを使用して表されます。 UTF-16 は、グループ 00 の平面 1 から 16 の UCS-4 コード位置間の変換と、特別なコードの組を定義するもので、Unicode 規格で定義された変換と同等です。

プログラミングのためのコンバーターの概要

各国語サポートには、データを (多くの場合) コード・セット間で変更できるグローバリゼーションのベースが備えられています。この目的のために、いくつかの標準コンバーターがサポートされています。

リモート・ホスト上に存在し、あるプログラムによって別のプログラムへ送信されたデータは、ソース・マシンのコード・セットから受信側のコード・セットへ変換しなければならない場合があります。 例えば、VM システムと通信している場合、ワークステーションは ISO8859-1 データを EBCDIC 形式に変換します。

コード・セットは、コード・ポイントに対して図形文字と制御文字の割り当てを定義します。 このようなコード化文字は、プログラムがあるコード・セットでデータを獲得し、別のコード・セットで表示する場合にも、変換する必要があります。

システムには、次の変換インターフェースが備えられています。

iconv コマンド
: *FromCode* および *ToCode* コード・セットを指定することにより、特定の変換を要求できるようになります。

libiconv 関数
: アプリケーションが名前でコンバーターを要求できるようになります。

システムは、ただちに使用できるコンバーターのライブラリーを備えています。 コンバーター・ライブラリーは、**/usr/lib/nls/loc/iconv/*** および **/usr/lib/nls/loc/iconvTable/*** ディレクトリーにあります。まったく必要ないのであれば、独自のコンバーターを定義しないでください。

コンバーター・ライブラリーには、コード・セット・コンバーターだけでなく、ネットワーク交換コンバーターも用意されています。 ネットワーク環境では、通信システムのコード・セットと通信のプロトコルにより、データを変換する方法が決まります。

交換コンバーターは、特定のシステムから別のシステムへ送信されるデータを変換するときに使用されます。 特定の内部コード・セットから別のコード・セットへの変換では、コード・セット・コンバーターが必要になります。 データを送信側のコード・セットから受信側のコード・セットへ変換するか、 8 ビット・データから 7 ビット・データに変換しなければならない場合、統一されたインターフェースが必要です。 **iconv** サブルーチンで、このインターフェースが実現します。

関連概念:

1 ページの『コード・セット間の変換』
文字とは、データの編成、制御、または表記に使用される記号のことです。特定言語の記述に使用されるこの種の記号のグループにより、文字セットが構成されます。 コード・セットには、文字セットのエンコード値が含まれます。 コード・セット内のエンコード値により、システムとその入出力デバイスとの間のインターフェースが設けられます。各国語サポートには、さまざまなコード・セット内にある文字エンコード値に従うコンバーターが備わっています。

2 ページの『コンバーターの概要』
グローバリゼーションには、データをコード・セット間で変更できるグローバリゼーションのベースが備えられています。テキスト・ファイルやメッセージ・カタログを変換する必要が生じることがあります。 この種の変換用に、複数の標準的なコンバーターがあります。

87 ページの『libiconv の理解』
このセクションでは、**iconv** アプリケーション・プログラミング・インターフェース (API) による変換について記載します。

iconv コマンドの使用

システムにインストールされているどのコンバーターも、 iconv ライブラリーを使用する iconv コマンドで使用できます。

iconv コマンドは、あるコード・セットから別のコード・セットへ変換するときのフィルターの役割を果たします。例えば、次のコマンドは、PC コード (IBM-850) から ISO8859-1 へのデータをフィルター操作します。

```
cat File | iconv -f IBM-850 -t ISO8859-1 | tftp -p - host /tmp/fo
```

iconv コマンドは、標準入力か指定ファイルのいずれかから読み取られた文字のエンコードを変換し、それからその結果を標準出力へ書き込みます。

注**:** AIX オペレーティング・システムでは、ISO10646-1 ラベルは UCS-2 エンコードを参照します。 このラベルは、UCS-2 の別名として使用できます。

libiconv の理解

このセクションでは、**iconv** アプリケーション・プログラミング・インターフェース (API) による変換について記載します。

多くの場合、**iconv** アプリケーション・プログラミング・インターフェース (API) は、変換を実現する以下のサブルーチンで構成されています。

iconv_open
: *FromCode* パラメーターによって指定されたコード・セットから *ToCode* パラメーターによって指定されたコード・セットへ文字を変換するときに必要な初期設定を行います。 指定されるストリングは、システムにインストールされているコンバーターによって異なります。 初期設定が成功したら、コンバーター・ディスクリプターの **iconv_t** が、初期状態に戻されます。

iconv
: **iconv_open** サブルーチンから入手した記述子を使用して、コンバーター関数を呼び出します。 *inbuf* キャラクターは入力バッファーの最初の文字を指し、 *inbytesleft* パラメーターは変換するバッファーの最後までのバイト数を示します。 *outbuf* パラメーターは出力バッファーの最初の使用可能バイトを指し、 *outbytesleft* パラメーターはバッファーの最後までの使用可能バイト数を示します。

  状態に依存するエンコードの場合、*inbuf* 値が NULL ポインターとなる呼び出しにより、サブルーチンは初期状態にされます。 続けて、NULL ポインター以外の呼び出しとして、*inbuf* パラメーターを指定して呼び出すと、機能の内部状態が必要に応じて変更されます。

iconv_close
: *cd* 変数によって指定された変換ディスクリプターをクローズして、再度使用できるようにします。

ネットワーク環境では、次の要因により、データの変換方法が決まります。

- 送信側と受信側のコード・セット
- 通信プロトコル (8 ビットまたは 7 ビット・データ)

次の表は、変換方法を要約し、異なる状態でデータを変換するお勧めの方法を紹介しています。

表 5. 同じコード・セットを使用するシステムとの通信

基準	通信プロトコル	通信プロトコル
選択する方法	7 ビットのみ	8 ビット
そのまま	無効	最善の選択
fold7	OK	OK
fold8	無効	OK
uucode	最善の選択	OK

表 6. 異なるコード・セットを使用したシステムとの通信 (または受信側のコード・セットが不明)

基準	通信プロトコル	通信プロトコル
選択する方法	7 ビットのみ	8 ビット
そのまま	無効	リモート・コード・セットが不明の場合は無効
fold7	最善の選択	OK
fold8	無効	最善の選択
uucode	無効	無効

送信側が受信側と同じコード・セットを使用する場合、次のような可能性が存在します。

- プロトコルで 8 ビット・データが許可されている場合、変換なしにデータを送信できます。
- プロトコルで 7 ビット・データだけが許可されている場合、 8 ビット・コード・ポイントを 7 ビット値にマップする必要があります。 **iconv** インターフェースと次のいずれかの方法を使用します。

メソッド	説明
uucode	**uuencode** および **uudecode** コマンドと同じようにマップされます。これは、お勧めの方法です。
7 ビット	7 ビット・データを使用して、内部コード・セットを変換します。 この方法では、変更なしに ASCII を渡します。

送信側が受信側とは異なるコード・セットを使用する場合、次の 2 つの可能性があります。

- プロトコルで 7 ビット・データだけが許可されている場合、fold7 の方法を使用します。
- プロトコルで 8 ビット・データが許可されていて、受信側のコード・セットを知っている場合、 iconv インターフェースを使用して、データを変換します。受信側のコード・セットを知らない場合、次の方法を使用します。

メソッド	説明
8 ビット	内部コード・セットを標準交換形式に変換します。 8 ビット・データが伝送され、受信側が自分のコード・セットでデータを再構築できるように、情報が保管されます。

関連概念:

85 ページの『プログラミングのためのコンバーターの概要』
各国語サポートには、データを (多くの場合) コード・セット間で変更できるグローバリゼーションのベースが備えられています。この目的のために、いくつかの標準コンバーターがサポートされています。

101 ページの『交換コンバーター - 7 ビット』
このコンバーターでは、内部コードと 7 ビットの標準交換形式 (fold7) の間の変換が実現します。

104 ページの『交換コンバーター - 8 ビット』
このコンバーターでは、内部コードと 8 ビットの標準交換形式 (fold8) の間の変換が実現します。
109 ページの『交換コンバーター — uucode』
このコンバーターは、 **uuencode** および **uudecode** コマンドと同じようにマップします。

iconv_open サブルーチンの使用

このセクションでは、さまざまな状態で **iconv_open** サブルーチンを使用する方法を示します。

以下の例には、さまざまな状態で **iconv_open** サブルーチンを使用する方法が示されています。

- 送信側と受信側が同じコード・セットを使用していて、プロトコルで 8 ビット・データが許可されている場合、データを変換せずに送信できます。プロトコルで 7 ビット・データだけが許可されている場合、次のようにします。

```
Sender:
 cd = iconv_open("uucode", nl_langinfo(CODESET));


Receiver:
 cd = iconv_open(nl_langinfo(CODESET), "uucode");
```

- 送信側と受信側が異なるコード・セットを使用している場合で、プロトコルで 8 ビット・データが許可されていて、受信側のコード・セットが不明の場合には、次のようにします。

```
Sender:
 cd = iconv_open("fold8", nl_langinfo(CODESET));


Receiver:
 cd = iconv_open(nl_langinfo(CODESET),"fold8" );
```

プロトコルで 7 ビット・データだけが許可されている場合、次のようにします。

```
Sender:
 cd = iconv_open("fold7", nl_langinfo(CODESET));


Receiver:
 cd = iconv_open(nl_langinfo(CODESET), "fold7" );
```

iconv_open サブルーチンでは、 **LOCPATH** 環境変数を使用して、名前が次の形式になっているコンバーターを探します。

iconv/*FromCodeSet_ToCodeSet*

FromCodeSet ストリングは送信側のコード・セットを表し、 *ToCodeSet* ストリングは受信側のコード・セットを表します。下線文字は、2 つのストリングを区切っています。

注**:** すべての **setuid** プログラムと **setgid** プログラムは、 **LOCPATH** 環境変数を無視します。

iconv コンバーターはロード可能なオブジェクト・モジュールであるため、 64 ビット環境で実行するときには、別のオブジェクトが必要になります。 64 ビット環境では、**iconv_open** ルーチンは、**LOCPATH** 環境変数を使用して、名前が次の形式になっているコンバーターを探します。

```
iconv/FromCodeSet_ToCodeSet__64.
```

iconv ライブラリーは、標準のコンバーター・オブジェクトをロードするか、 64 ビットのコンバーター・オブジェクトをロードするかを自動的に選択します。 **iconv_open** サブルーチンがコンバーターを検出しない場合、 **from**、**to** のペアを使用して、テーブルに基づく変換を定義したファイルを探します。 このファイルには、**genxlt** コマンド で作成された変換テーブルが含まれています。

iconvTable コンバーターは、**LOCPATH** 環境変数を使用して、名前が次の形式になっているファイルを探します。

iconvTable/*FromCodeSet_ToCodeSet*

コンバーターが見つかったら、ロード操作を実行して初期設定されます。 コンバーター・ディスクリプターの **iconv_t** は、初期状態に戻されます。

コンバーター・プログラム対コンバーター・テーブル

コンバーター・プログラムは、一連の規則に沿ってデータを変換する実行可能な機能です。 コンバーター・テーブルは、ステートレス変換を実行する単一バイト変換テーブルです。

プログラムおよびテーブルは、次のような別のディレクトリーにあります。

ディレクトリー	説明
/usr/lib/nls/loc/iconv	コンバーター・プログラム
/usr/lib/nls/loc/iconvTable	コンバーター・テーブル

コンバーター・プログラムをコンパイルして **libiconv.a** ライブラリーにリンクしたら、プログラムは **/usr/lib/nls/loc/iconv** ディレクトリーに置かれます。

テーブル・コンバーターを作成するには、ソース・コンバーター・テーブル・ファイルを作成します。変換テーブルをテーブル・コンバーターが理解できる形式にコンパイルするには、 **genxlt** コマンドを使用します。そうすると、出力ファイルが **/usr/lib/nls/loc/iconvTable** ディレクトリーに置かれます。

ユニコード・コンバーターとユニバーサル・コンバーター

このセクションでは、ユニコード変換テーブルとユニバーサル・コンバーター・プログラムについて記載します。

Unicode (または UCS-2) 変換テーブルは、次の場所にあります。

```
$LOCPATH/uconvTable/*CodeSet*
```

$LOCPATH/uconv/UCSTBL コンバーター・プログラムは、**iconv** ユーティリティーを使用して、UCS-2 へのまたは UCS-2 からの変換を実行するときに使用されます。

UCS-2 への変換と UCS-2 からの変換が定義されている任意の 2 つのコード・セット間での変換に使用できる、ユニバーサル・コンバーター・プログラムが提供されています。次のような uconv テーブルがあります。

```
X     -> UCS-2
UCS-2 -> Y
```

次のようにマップするユニバーサル変換を定義できます。

```
X -> UCS-2 -> Y
```

このときに、**$LOCPATH/iconv/Universal_UCS_Conv** を使用します。

ユニバーサル UCS コンバーター

UCS-2 は、ユニバーサル 16 ビット・エンコードであり、仮想的な任意のコード・セット間での変換機能を備えた交換手段として使用できます。

変換は、ユニバーサル UCS コンバーターを使用して実現できます。これは、次のように、任意の 2 つのコード・セット XXX と YYY との間で変換を行います。

```
XXX <-> UTF-32 <-> YYY
```

XXX と YYY の変換は、サポートされている UCS-2 交換コンバーターのリストに含まれていなければならず、システムにインストールされていなければなりません。

ユニバーサル・コンバーターは、ファイル **/usr/lib/nls/loc/iconv/Universal_UCS_Conv** としてインストールされています。

マルチバイト・コードとワイド文字コードの間の変換は、現行ロケールの設定に応じて異なります。 使用しようとしている個々のロケールが一貫性のある方法でワイド文字コードを処理することが分かっていない場合は、 2 つのプロセス間でワイド文字コードを交換しないでください。 このオペレーティング・システム用のほとんどのロケールでは、ワイド文字コードとして、Unicode 文字値を使用しています。ただし、IBM-eucTW コード・セットに基づくロケールの場合は除きます。

コンバーターの使用

このセクションでは、iconv インターフェース・サブルーチンについて記載します。

iconv インターフェースは、変換をオープン、実行、およびクローズするときに使用される、次のサブルーチンを 1 つにまとめたものです。

- **iconv_open**
- **iconv**
- **iconv_close**

コード・セット変換フィルターの例

このセクションでは、*ToCode* パラメーターと *FromCode* パラメーターを受け入れるコード・セット変換フィルターについて記載します。

次の例には、入力引数として *ToCode* および *FromCode* パラメーターを受け入れるコード・セット変換フィルターを作成するために、これらのサブルーチンを使用する方法が示されています。

```
#include <stdio.h>
#include <nl_types.h>
#include <iconv.h>
#include <string.h>
#include <errno.h>
#include <locale.h>

#define ICONV_DONE() (r>=0)
#define ICONV_INVAL() (r<0) && (errno==EILSEQ))
#define ICONV_OVER() (r<0) && (errno==E2BIG))
#define ICONV_TRUNC() (r<0) && (errno==EINVAL))

#define USAGE 1
#define ERROR 2
#define INCOMP 3

char ibuf[BUFSIZ], obuf[BUFSIZ];

extern int errno;

main (argc,argv)
int argc;
char **argv;
{
```

```
 size_t  ileft,oleft;
 nl_catd catd;
 iconv_t cd;
 int r;
 char *ip,*op;

 setlocale(LC_ALL,"");
 catd = catopen (argv[0],0);

 if(argc!=3){
  fprintf(stderr,
   catgets (catd,NL_SETD,USAGE,"usage;conv fromcode tocode¥n"));
  exit(1);
 }

 cd=iconv_open(argv[2],argv[1]);

ileft=0;

while(!feof(stdin)) {

 /*
 * After the next operation,ibuf will
 * contain new data plus any truncated
 * data left from the previous read.
 */
 ileft+=fread(ibuf+ileft,1,BUFSIZ-ileft,stdin);
 do {
  ip=ibuf;
  op=obuf;
  oleft=BUFSIZ;

  r=iconv(cd,&ip,&ileft,&op,&oleft);

  if(ICONV_INVAL()){
   fprintf(stderr,
      catgets(catd,NL_SETD,ERROR,"invalid input¥n"));
   exit(2);
  }

 fwrite(obuf,1,BUFSIZ-oleft,stdout);

 if(ICONV_TRUNC() || ICONV_OVER())
  /*
  *Data remaining in buffer-copy
  *it to the beginning
  */

  memcpy(ibuf,ip,ileft);

  /*
  *loop until all characters in the input
  *buffer have been converted.
  */
 } while(ICONV_OVER());
}

 if(ileft!=0){
  /*
  *This can only happen if the last call
  *to iconv() returned ICONV_TRUNC, meaning
  *the last data in the input stream was
  *incomplete.
  */
 fprintf(stderr,catgets(catd,NL_SETD,INCOMP,"input incomplete¥n"));
 exit(3);
 }
```

```
 iconv_close(cd);
 exit(0);
}
```

コンバーターの命名

このセクションでは、コード・セット名について記載します。

コード・セット名は、*CodesetRegistry*-*CodesetEncoding* の形式になります。

コード・セットの形式	説明
CodesetRegistry	エンコードの登録権限を識別します。 *CodesetRegistry* は、移植可能コード・セットからの文字 (通常は A-Z と 0-9) で構成されていなければなりません。
CodesetEncoding	登録された権限で定義されたコード化文字セットを識別します。

iconv コマンドと **iconv_open** サブルーチンで使用される from,to 変数は、名前の形式が */usr/lib/nls/loc/iconv/%f_%t* または */usr/lib/nls/loc/iconvTable/%f_%t* であるべきファイルを識別します。ここで、

変数	説明
%f	*FromCode* セット名を表します。
%t	*ToCode* セット名を表します。

コンバーターのリスト

コンバーターは、データをあるコード・セットから別のコード・セットに変更します。 次のセクションでは、iconv ライブラリーでサポートされているコンバーター群がリストされています。

BOS Runtime Environment に付属するコンバーターはすべて、 **/usr/lib/nls/loc/iconv/*** または **/usr/lib/nls/loc/iconvTable/*** ディレクトリーにあります。

これらのディレクトリーには、専用 コンバーター、つまり、他のコンバーターによって使用されるコンバーターも含まれます。 しかし、ユーザーとプログラムは、次のリストのコンバーターにのみ依存しています。

BOS Runtime Environment に付属するコンバーターで、このリストに載せられていないものは、専用コンバーターであり、変更されるか削除されることになります。 他の製品で提供されるコンバーターは、**/usr/lib/nls/loc/iconv/*** または **/usr/lib/nls/loc/iconvTable/*** ディレクトリーに置くことができます。

プログラマーは、登録されたコード・セット名か、アプリケーションに関連付けられたコード・セット名を使用することが勧められています。 X Consortium は、参照用にコード・セット名の登録を保管しています。

関連概念:

50 ページの『各国語サポートのコード・セット』

AIX のグローバリゼーションは、すべてのコード・セットは任意の数の文字セットに分けられるという仮定に基づいています。

PC、ISO、および EBCDIC のコード・セット・コンバーター

これらのコンバーターでは、PC、ISO、および EBCDIC 単一バイト・ステートレス・コード・セット間での変換を実現します。 次のタイプの変換がサポートされています。PC と ISO 間、PC と EBCDIC 間、および ISO と EBCDIC 間。

Latin-1 から Latin-1 やギリシャ語からギリシャ語など、互換性のあるコード・セット間で変換が行えます。 しかし、異なる EBCDIC 国コード・セット間での変換はサポートされていません。 非互換の文字セット間の変換については、 101 ページの『交換コンバーター - 7 ビット』および 104 ページの『交換コンバーター - 8 ビット』を参照してください。

iconvTable ディレクトリーの変換テーブルは、 **genxlt** コマンドで作成されます。

関連概念:

101 ページの『交換コンバーター - 7 ビット』
このコンバーターでは、内部コードと 7 ビットの標準交換形式 (fold7) の間の変換が実現します。

104 ページの『交換コンバーター - 8 ビット』
このコンバーターでは、内部コードと 8 ビットの標準交換形式 (fold8) の間の変換が実現します。

関連情報:

genxlt コマンド

互換コード・セット名

次の表は、互換性のあるコード・セット名をリストしています。それぞれの行では、コンバーターを要求するときに使用できる to/from ストリングを定義しています。

注: PC および ISO コード・セットは ASCII ベースです。

表 7. コード・セットの互換性

文字セット	言語	PC	ISO	EBCDIC
Latin-1	米国英語、ポルトガル語、カナダ・フランス語	N/A	ISO8859-1	IBM-037
Latin-1	デンマーク語、ノルウェー語	N/A	ISO8859-1	IBM-277
Latin-1	フィンランド語、スウェーデン語	N/A	ISO8859-1	IBM-278
Latin-1	イタリア語	N/A	ISO8859-1	IBM-280
Latin-1	日本語	N/A	ISO8859-1	IBM-281
Latin-1	スペイン語	N/A	ISO8859-1	IBM-284
Latin-1	英国英語	N/A	ISO8859-1	IBM-285
Latin-1	ドイツ語	N/A	ISO8859-1	IBM-273
Latin-1	フランス語	N/A	ISO8859-1	IBM-297
Latin-1	ベルギー語、スイス・ドイツ語	N/A	ISO8859-1	IBM-500
Latin-2	クロアチア語、チェコスロバキア語、ハンガリー語、ポーランド語、ルーマニア語、セルビア語系ラテン語、スロバキア語、スロベニア語	IBM-852	ISO88859-2	IBM-870

表 7. コード・セットの互換性 (続き)

文字セット	言語	PC	ISO	EBCDIC
キリル文字	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語	IBM-855	ISO8859-5	IBM-880 IBM-1025
キリル文字	ロシア語	IBM-866	ISO8859-5	IBM-1025
ヘブライ語	ヘブライ語	IBM-856 IBM-862	ISO8859-8	IBM-424 IBM-803
トルコ語	トルコ語	IBM-857	ISO8859-9	IBM-1026
アラビア語	アラビア語	IBM-864 IBM-1046	ISO8859-6	IBM-420
ギリシャ語	ギリシャ語	IBM-869	ISO8859-7	IBM-875
ギリシャ語	ギリシャ語	IBM-869	ISO8859-7	IBM-875
バルト語	リトアニア語、ラトビア語、エストニア語	IBM-921 IBM-922	ISO8859-4	IBM-1112 IBM-1122

注: ソース・コード・セットに存在しても、ターゲット・コード・セットに存在しない文字は、コンバーターが定義した置換文字に変換されます。

ファイル

この表では、**/usr/lib/nls/loc/iconvTable** ディレクトリーにある **inconvTable** コンバーターについて説明しています。

表 8. *iconvTable* コンバーター

コンバーター・テーブル	説明	言語
IBM-037_IBM-850	IBM-037 から IBM-850	米国英語、ポルトガル語、カナダ・フランス語
IBM-273_IBM-850	IBM-273 から IBM-850	ドイツ語
IBM-277_IBM-850	IBM-277 から IBM-850	デンマーク語、ノルウェー語
IBM-278_IBM-850	IBM-278 から IBM-850	フィンランド語、スウェーデン語
IBM-280_IBM-850	IBM-280 から IBM-850	イタリア語
IBM-281_IBM-850	IBM-281 から IBM-850	日本語英数小文字
IBM-284_IBM-850	IBM-284 から IBM-850	スペイン語
IBM-285_IBM-850	IBM-285 から IBM-850	英国英語
IBM-297_IBM-850	IBM-297 から IBM-850	フランス語
IBM-420_IBM_1046	IBM-420 から IBM-1046	アラビア語
IBM-424_IBM-856	IBM-424 から IBM-856	ヘブライ語
IBM-424_IBM-862	IBM-424 から IBM-862	ヘブライ語
IBM-500_IBM-850	IBM-500 から IBM-850	ベルギー語、スイス・ドイツ語
IBM-803_IBM-856	IBM-803 から IBM-856	ヘブライ語
IBM-803_IBM-862	IBM-803 から IBM-862	ヘブライ語
IBM-850_IBM-037	IBM-850 から IBM-037	米国英語、ポルトガル語、カナダ・フランス語
IBM-850_IBM-273	IBM-850 から IBM-273	ドイツ語
IBM-850_IBM-277	IBM-850 から IBM-277	デンマーク語、ノルウェー語
IBM-850_IBM-278	IBM-850 から IBM-278	フィンランド語、スウェーデン語
IBM-850_IBM-280	IBM-850 から IBM-280	イタリア語
IBM-850_IBM-281	IBM-850 から IBM-281	日本語英数小文字
IBM-850_IBM-284	IBM-850 から IBM-284	スペイン語
IBM-850_IBM-285	IBM-850 から IBM-285	英国英語

表 8. *iconvTable* コンバーター (続き)

コンバーター・テーブル	説明	言語
IBM-850_IBM-297	IBM-850 から IBM-297	フランス語
IBM-850_IBM-500	IBM-850 から IBM-500	ベルギー語、スイス・ドイツ語
IBM-856_IBM-424	IBM-856 から IBM-424	ヘブライ語
IBM-856_IBM-803	IBM-856 から IBM-803	ヘブライ語
IBM-856_IBM-862	IBM-856 から IBM-862	ヘブライ語
IBM-862_IBM-424	IBM-862 から IBM-424	ヘブライ語
IBM-862_IBM-803	IBM-862 から IBM-803	ヘブライ語
IBM-862_IBM-856	IBM-862 から IBM-856	ヘブライ語
IBM-864_IBM-1046	IBM-864 から IBM-1046	アラビア語
IBM-921_IBM-1112	IBM-921 から IBM-1112	リトアニア語、ラトビア語
IBM-922_IBM-1122	IBM-922 から IBM-1122	エストニア語
IBM-1112_IBM-921	IBM-1121 から IBM-921	リトアニア語、ラトビア語
IBM-1122_IBM-922	IBM-1122 から IBM-922	エストニア語
IBM-1046_IBM-420	IBM-1046 から IBM-420	アラビア語
IBM-1046_IBM-864	IBM-1046 から IBM-864	アラビア語
IBM-037_ISO8859-1	IBM-037 から ISO8859-1	米国英語、ポルトガル語、カナダ・フランス語
IBM-273_ISO8859-1	IBM-273 から ISO8859-1	ドイツ語
IBM-277_ISO8859-1	IBM-277 から ISO8859-1	デンマーク語、ノルウェー語
IBM-278_ISO8859-1	IBM-278 から ISO8859-1	フィンランド語、スウェーデン語
IBM-280_ISO8859-1	IBM-280 から ISO8859-1	イタリア語
IBM-281_ISO8859-1	IBM-281 から ISO8859-1	日本語英数小文字
IBM-284_ISO8859-1	IBM-284 から ISO8859-1	スペイン語
IBM-285_ISO8859-1	IBM-285 から ISO8859-1	英国英語
IBM-297_ISO8859-1	IBM-297 から ISO8859-1	フランス語
IBM-420_ISO8859-6	IBM-420 から ISO8859-6	アラビア語
IBM-424_ISO8859-8	IBM-424 から ISO8859-8	ヘブライ語
IBM-500_ISO8859-1	IBM-500 から ISO8859-1	ベルギー語、スイス・ドイツ語
IBM-803_ISO8859-8	IBM-803 から ISO8859-8	ヘブライ語
IBM-852_ISO8859-2	IBM-852 から ISO8859-2	クロアチア語、チェコスロバキア語、ハンガリー語、ポーランド語、ルーマニア語、セルビア語系ラテン語、スロバキア語、スロベニア語
IBM-855_ISO8859-5	IBM-855 から ISO8859-5	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語
IBM-866_ISO8859-5	IBM-866 から ISO8859-5	ロシア語
IBM-869_ISO8859-7	IBM-869 から ISO8859-7	ギリシャ語
IBM-875_ISO8859-7	IBM-875 から ISO8859-7	ギリシャ語
IBM-870_ISO8859-2	IBM-870 から ISO8859-2	クロアチア語、チェコスロバキア語、ハンガリー語、ポーランド語、ルーマニア語、セルビア語、スロバキア語、スロベニア語
IBM-880_ISO8859-5	IBM-880 から ISO8859-5	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語
IBM-1025_ISO8859-5	IBM-1025 から ISO8859-5	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語
IBM-857_ISO8859-9	IBM-857 から ISO8859-9	トルコ語

表 8. *iconvTable* コンバーター (続き)

コンバーター・テーブル	説明	言語
IBM-1026_ISO8859-9	IBM-1026 から ISO8859-9	トルコ語
IBM-850_ISO8859-1	IBM-850 から ISO8859-1	ラテン語
IBM-856_ISO8859-8	IBM-856 から ISO8859-8	ヘブライ語
IBM-862_ISO8859-8	IBM-862 から ISO8859-8	ヘブライ語
IBM-864_ISO8859-6	IBM-864 から ISO8859-6	アラビア語
IBM-1046_ISO8859-6	IBM-1046 から ISO8859-6	アラビア語
ISO8859-1_IBM-850	ISO8859-1 から IBM-850	ラテン語
ISO8859-6_IBM-864	ISO8859-6 から IBM-864	アラビア語
ISO8859-6_IBM-1046	ISO8859-6 から IBM-1046	アラビア語
ISO8859-8_IBM-856	ISO8859-8 から IBM-856	ヘブライ語
ISO8859-8_IBM-862	ISO8859-8 から IBM-862	ヘブライ語
ISO8859-1_IBM-037	ISO8859-1 から IBM-037	米国英語、ポルトガル語、カナダ・フランス語
ISO8859-1_IBM-273	ISO8859-1 から IBM-273	ドイツ語
ISO8859-1_IBM-277	ISO8859-1 から IBM-277	デンマーク語、ノルウェー語
ISO8859-1_IBM-278	ISO8859-1 から IBM-278	フィンランド語、スウェーデン語
ISO8859-1_IBM-280	ISO8859-1 から IBM-280	イタリア語
ISO8859-1_IBM-281	ISO8859-1 から IBM-281	日本語英数小文字
ISO8859-1_IBM-284	ISO8859-1 から IBM-284	スペイン語
ISO8859-1_IBM-285	ISO8859-1 から IBM-285	英国英語
ISO8859-1_IBM-297	ISO8859-1 から IBM-297	フランス語
ISO8859-1_IBM-500	ISO8859-1 から IBM-500	ベルギー語、スイス・ドイツ語
ISO8859-2_IBM-852	ISO8859-2 から IBM-852	クロアチア語、チェコスロバキア語、ハンガリー語、ポーランド語、ルーマニア語、セルビア語系ラテン語、スロバキア語、スロベニア語
ISO8859-2_IBM-870	ISO8859-2 から IBM-870	クロアチア語、チェコスロバキア語、ハンガリー語、ポーランド語、ルーマニア語、セルビア語系ラテン語、スロバキア語、スロベニア語
ISO8859-5_IBM-855	ISO8859-5 から IBM-855	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語
ISO8859-5_IBM-880	ISO8859-5 から IBM-880	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語
ISO8859-5_IBM-1025	ISO8859-5 から IBM-1025	ブルガリア語、マケドニア語、セルビア語キリル文字、ロシア語
ISO8859-6_IBM-420	ISO8859-6 から IBM-420	アラビア語
ISO8859-5_IBM-866	ISO8859-5 から IBM-866	ロシア語
ISO8859-7_IBM-869	ISO8859-7 から IBM-869	ギリシャ語
ISO8859-7_IBM-875	ISO8859-7 から IBM-875	ギリシャ語
ISO8859-8_IBM-424	ISO8859-8 から IBM-424	ヘブライ語
ISO8859-8_IBM-803	ISO8859-8 から IBM-803	ヘブライ語
ISO8859-9_IBM-857	ISO8859-9 から IBM-857	トルコ語
ISO8859-9_IBM-1026	ISO8859-9 から IBM-1026	トルコ語

マルチバイト・コード・セット・コンバーター

このセクションでは、マルチバイト・コード・セット・コンバーターが変換に使用するコード・セットについて説明します。

マルチバイト・コード・セット・コンバーターは、以下のコード・セットの間で文字を変換します。

- PC マルチバイト・コード・セット
- EUC マルチバイト・コード・セット (ISO ベース)
- EBCDIC マルチバイト・コード・セット

次の表は、互換性のあるコード・セット名をリストしています。 それぞれの行では、コンバーターを要求するときに使用できる to/from ストリングを定義しています。

表 *9.* コード・セットの互換性

言語	**PC**	**ISO**	**EBCDIC**
日本語	IBM-932	IBM-eucJP	IBM-930、IBM-939
日本語 (MS 互換)	IBM-943	IBM-eucJP	IBM-930、IBM-939
韓国語	IBM-934	IBM-eucKR	IBM-933
中国語 (繁体字)	IBM-938、big-5	IBM-eucTW	IBM-937
中国語 (簡体字)	IBM-1381	IBM-eucCN	IBM-935

1. 中国語 (簡体字) と中国語 (繁体字) の間で変換を行うことができます (IBM-eucTW <—> IBM-eucCN と big5 <—> IBM-eucCN)。
2. UTF-8 は、追加のコード・セットです。

関連概念:

112 ページの『UTF-8 交換コンバーター』

このセクションでは、各コード・セットと UTF-8 の両方向に行われる変換について説明します。

ファイル

次のリストでは、**/usr/lib/nls/loc/iconv** ディレクトリーにあるマルチバイト・コード・セット・コンバーターについて説明します。

コンバーター	説明
IBM-eucJP_IBM-932	IBM-eucJP から IBM-932
IBM-eucJP_IBM-943	IBM-eucJP から IBM-943
IBM-eucJP_IBM-930	IBM-eucJP から IBM-930
IBM-eucCN_IBM-936(PC5550)	IBM-eucCN から IBM-936(PC5550)
IBM-eucCN_IBM-935	IBM-eucCN から IBM-935
IBM-eucJP_IBM-939	IBM-eucJP から IBM-939
IBM-eucCN_IBM-1381	IBM-eucCN から IBM-1381
IBM-943_IBM-932	IBM-943 から IBM-932
IBM-932_IBM-943	IBM-932 から IBM-943
IBM-930_IBM-932	IBM-930 から IBM-932
IBM-930_IBM-943	IBM-930 から IBM-943
IBM-930_IBM-eucJP	IBM-930 から IBM-eucJP
IBM-932_IBM-eucJP	IBM-932 から IBM-eucJP
IBM-932_IBM-930	IBM-932 から IBM-930
IBM-943_IBM-eucJP	IBM-943 から IBM-eucJP

コンバーター	説明
IBM-943_IBM-930	IBM-943 から IBM-930
IBM-936(PC5550)_IBM-935	IBM-936(PC5550) から IBM-935
IBM-936_IBM-935	IBM-936 から IBM-935
IBM-932_IBM-939	IBM-932 から IBM-939
IBM-939_IBM-932	IBM-939 から IBM-932
IBM-943_IBM-939	IBM-943 から IBM-939
IBM-939_IBM-943	IBM-939 から IBM-943
IBM-935_IBM-936(PC5550)	IBM-935 から IBM-936(PC5550)
IBM-935_IBM-936	IBM-935 から IBM-936
IBM-1381_IBM-935	IBM-1381 から IBM-935
IBM-935_IBM-1381	IBM-935 から IBM-1381
IBM-935_IBM-eucCN	IBM-935 から IBM-eucCN
IBM-936(PC5550)_IBM-eucCN	IBM-936(PC5550) から IBM-eucCN
IBM-eucTW_IBM-eucCN	IBM-eucTW から IBM-eucCN
big5_IBM-eucCN	big5 から IBM-eucCN
IBM-1381_IBM-eucCN	IBM-1381 から IBM-eucCN
IBM-939_IBM-eucJP	IBM-939 から IBM-eucJP
IBM-eucKR_IBM-934	IBM-eucKR から IBM-934
IBM-934_IBM-eucKR	IBM-934 から IBM-eucKR
IBM-eucKR_IBM-933	IBM-eucKR から IBM-933
IBM-933_IBM-eucKR	IBM-933 から IBM-eucKR
IBM-eucTW_IBM-937	IBM-eucTW から IBM-937
IBM-938_IBM-937	IBM-938 から IBM-937
big-5_IBM-937	big-5 から IBM-937
IBM-eucCN_IBM-eucTW	IBM-eucCN から IBM-eucTW
IBM-937_IBM-eucTW	IBM-937 から IBM-eucTW
IBM-937_IBM-938	IBM-937 から IBM-938
IBM-eucTW_IBM-938	IBM-eucTW から IBM-938
IBM-eucCN_big5	IBM-eucCN から big5
IBM-eucTW_big-5	IBM-eucTW から big-5
IBM-937_big-5	IBM-937 から big-5
CNS11643.1992-3_IBM-eucTW	CNS11643.1992-3 から IBM-eucTW
CNS11643.1992-3-GL_IBM-eucTW	CNS11643.1992-3-GL から IBM-eucTW
CNS11643.1992-3-GR_IBM-eucTW	CNS11643.1992-3-GR から IBM-eucTW
CNS11643.1992-4_IBM-eucTW	CNS11643.1992-4 から IBM-eucTW
CNS11643.1992-4-GL_IBM-eucTW	CNS11643.1992-4-GL から IBM-eucTW
CNS11643.1992-4-GR_IBM-eucTW	CNS11643.1992-4-GR から IBM-eucTW
IBM-eucTW_CNS11643.1992-3	IBM-eucTW から CNS11643.1992-3
IBM-eucTW_CNS11643.1992-3-GL	IBM-eucTW から CNS11643.1992-3-GL
IBM-eucTW_CNS11643.1992-3-GR	IBM-eucTW から CNS11643.1992-3-GR
IBM-eucTW_CNS11643.1992-4	IBM-eucTW から CNS11643.1992-4
IBM-eucTW_CNS11643.1992-4-GL	IBM-eucTW から CNS11643.1992-4-GL
IBM-eucTW_CNS11643.1992-4-GR	IBM-eucTW から CNS11643.1992-4-GR
IBM-eucCN_GB2312.1980-1	IBM-eucCN から GB2312.1980-1
IBM-eucCN_GB2312.1980-1-GL	IBM-eucCN から GB2312.1980-1-GL
IBM-eucCN_GB2312.1980-1-GR	IBM-eucCN から GB2312.1980-1-GR

コンバーター	説明
IBM-937_csic	IBM-937 から csic
csic_IBM-937	csic から IBM-937
IBM-938_csic	IBM-938 から csic
csic_IBM-938	csic から IBM-938
IBM-eucTW_ccdc	IBM-eucTW から ccdc
ccdc_IBM-eucTW	ccdc から IBM-eucTW
IBM-eucTW_cns	IBM-eucTW から cns
cns_IBM-eucTW	cns から IBM-eucTW
IBM-eucTW_csic	IBM-eucTW から csic
csic_IBM-eucTW	csic から IBM-eucTW
IBM-eucTW_sops	IBM-ecuTW から sops
sops_IBM-eucTW	sops から IBM-eucTW
IBM-eucTW_tca	IBM-eucTW から tca
tca_IBM-eucTW	tca から IBM-eucTW
big5_cns	big5 から cns
cns_big5	cns から big5
big5_csic	big5 から csic
csic_big5	csic から big5
big5_ttc	big5 から ttc
ttc_big5	ttc から big5
big5_ttcmin	big5 から ttcmin
ttcmin_big5	ttcmin から big5
big5_unicode	big5 から unicode
unicode_big5	unicode から big5
big5_wang	big5 から wang
wang_big5	wang から big5
ccdc_csic	ccdc から csic
csic_ccdc	csic から ccdc
csic_sops	csic から sops
sops_csic	sops から csic
CNS11643.1986-1_big5	CNS11643.1986-1 から big5
big5_CNS11643.1986-1	big5 から CNS11643.1986-1
CNS11643.1986-1-GR_big5	CNS11643.1986-1-GR から big5
big5_CNS11643.1986-1-GR	big5 から CNS11643.1986-1-GR
CNS11643.1986-2_big5	CNS11643.1986-2 から big5
big5_CNS11643.1986-2	big5 から CNS11643.1986-2
CNS11643.1986-2-GR_big5	CNS11643.1986-2-GR から big5
big5_CNS11643.1986-2-GR	big5 から CNS11643.1986-2-GR
CNS11643.CT-GR_big5	CNS11643.CT-GR から big5
big5_CNS11643.CT-GR	big5 から CNS11643.CT-GR
IBM-sbdTW-GR_big5	IBM-sbdTW-GR から big5
big5_IBM-sbdTW-GR	big5 から IBM-sbdTW-GR
IBM-sbdTW.CT-GR_big5	IBM-sbdTW.CT-GR から big5
big5_IBM-sbdTW.CT-GR	big5 から IBM-sbdTW.CT-GR
IBM-sbdTW_big5	IBM-sbdTW から big5
big5_IBM-sbdTW	big5 から IBM-sbdTW

コンバーター	説明
IBM-udcTW-GR_big5	IBM-udcTW-GR から big5
big5_IBM-udcTW-GR	big5 から IBM-udcTW-GR
IBM-udcTW.CT-GR_big5	IBM-udcTW.CT-GR から big5
big5_IBM-udcTW.CT-GR	big5 から IBM-udcTW.CT-GR
ISO8859-1_big5	ISO8859 から big5
big5_ISO8859-1	big5 から ISO8859-1
IBM-sbdTW_big5	IBM-sbdTW から big5
big5_IBM-sbdTW	big5 から IBM-sbdTW
big5_ASCII-GR	big5 から ASCII-GR
ASCII-GR_big5	ASCII-GR から big5
GBK_big5	GBK から big5
big5_GBK	big5 から GBK
GBK_IBM-eucTW	GBK から IBM-eucTW
IBM-eucTW_GBK	IBM-eucTW から GBK
CNS11643.1986-1_GBK	CNS11643.1986-1 から GBK
GBK_CNS11643.1986-1	GBK から CNS11643.1986-1
CNS11643.1986-2_GBK	CNS11643.1986-2 から GBK
GBK_CNS11643.1986-2	GBK から CNS11643.1986-2
CNS11643.1986-1-GR_GBK	CNS11643.1986-1-GR から GBK
GBK_CNS11643.1986-1-GR	GBK から CNS11643.1986-1-GR
CNS11643.1986-2-GR_GBK	CNS11643.1986-2-GR から GBK
GBK_CNS11643.1986-2-GR	GBK から CNS11643.1986-2-GR
CNS11643.1986-1-GL_GBK	CNS11643.1986-1-GL から GBK
GBK_CNS11643.1986-1-GL	GBK から CNS11643.1986-1-GL
CNS11643.1986-2-GL_GBK	CNS11643.1986-2-GL から GBK
GBK_CNS11643.1986-2-GL	GBK から CNS11643.1986-2-GL
CNS11643.CT-GR_GBK	CNS11643.CT-GR から GBK
GBK_CNS11643.CT-GR	GBK から CNS11643.CT-GR
GB2312.1980.CT-GR_GBK	GB2312.1980.CT-GR から GBK
GBK_GB2312.1980.CT-GR	GBK から GB2312.1980.CT-GR
GB2312.1980-0_GBK	GBK2312.1980-0 から GBK
GBK_GB2312.1980-0	GBK から GBK2312.1980-0
GB2312.1980-0-GR_GBK	GB2312.1980-0-GR から GBK
GBK_GB2312.1980-0-GR	GBK から GB2312.1980-0-GR
GB2312.1980-0-GL_GBK	GB2312.1980-0-GL から GBK
GBK_GB2312.1980-0-GL	GBK から GB2312.1980-0-GL
ASCII-GR_GBK	ASCII-GR から GBK
GBK_ASCII-GR	GBK から ASCII-GR
ISO8859-1_GBK	ISO8859-1 から GBK
GBK_ISO8859-1	GBK から ISO8859-1
IBM-eucCN_GBK	IBM-eucCN から GBK
GBK_IBM-eucCN	GBK から IBM-eucCN

交換コンバーター - 7 ビット

このコンバーターでは、内部コードと 7 ビットの標準交換形式 (fold7) の間の変換が実現します。

fold7 名は、7 ビット・メール・プロトコルを使用してテキスト・データを渡すのに使用できるエンコードを示します。 これらのエンコードは、ISO2022 に基づいています。

fold7 コンバーターは、コード・セットからの文字を、それぞれの文字を示す標準 7 ビット・エンコードに変換します。 このタイプの変換は、クライアントが、別のコード・セットと通信しているが、同じ文字セットを使用するようなネットワークで役立ちます。 例えば、以下のように使います。

コード・セット変換	説明
IBM-850 <—> ISO8859-1	共通ラテン語文字
IBM-932 <—> IBM-eucJP	共通日本語文字

以下のエスケープ・シーケンスは、標準のコード・セットを指定します。

エスケープ・シーケンス	標準コード・セット
01/11 02/04 04/00	GL JIS X0208.1978-0。
01/11 02/04 02/08 04/01	GB2312.1980-0 の GL 左半分。
01/11 02/08 04/02	GL 7 ビット ASCII または ISO8859-1 の左半分。
01/11 02/14 04/01	ISO8859-1 の GL 右半分。
01/11 02/14 04/02	ISO8859-2 の GL 右半分。
01/11 02/14 04/03	ISO8859-3 の GL 右半分。
01/11 02/14 04/04	ISO8859-4 の GL 右半分。
01/11 02/14 04/06	ISO8859-7 の GL 右半分。
01/11 02/14 04/07	ISO8859-6 の GL 右半分。
01/11 02/14 04/08	ISO8859-8 の GL 右半分。
01/11 02/14 04/12	ISO8859-5 の GL 右半分。
01/11 02/14 04/13	ISO8859-9 の GL 右半分。
01/11 02/08 04/09	JIS X0201.1976-0 の GL 右半分。
01/11 02/08 04/10	JIS X0201.1976 の GL 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 JIS X0208.1978-0。
01/11 02/05 02/15 03/01 M L 06/09 06/02 06/13 02/13 03/08 03/05 03/00 00/02	IBM-850 固有文字の GL 右半分。 ISO8859-1 と共通の文字では、このエスケープ・シーケンスは使用しません。
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 日本語 (IBM-udcJP) ユーザー定義可能文字。
01/11 02/04 02/08 04/03	GL KSC5601-1987。
01/11 02/04 02/09 03/00	GL CNS11643-1986-1。
01/11 02/04 02/10 03/01	GL CNS11643-1986-2。
01/11 02/05 02/15 03/00 M L 05/05 05/04 04/06 02/13 03/07 00/02	base64 としてエンコードされた UCS-2。上にリストされた他のいずれかの 7 ビット・エスケープ・シーケンスでエンコードされていない文字の場合にのみ使用されます。

あるコード・セットから fold7 に変換する場合、コード・セットを指定するのに使用されるエスケープ・シーケンスは、リストされた順序で選択されます。例えば、JISX0208.1983-0 文字は、**01/11 01/04 04/02** を指定します。

関連概念:

87 ページの『libiconv の理解』
このセクションでは、**iconv** アプリケーション・プログラミング・インターフェース (API) による変換に

ついて記載します。

94 ページの『PC、ISO、および EBCDIC のコード・セット・コンバーター』
これらのコンバーターでは、PC、ISO、および EBCDIC 単一バイト・ステートレス・コード・セット間での変換を実現します。 次のタイプの変換がサポートされています。PC と ISO 間、PC と EBCDIC 間、および ISO と EBCDIC 間。

ファイル

このリストでは、**/usr/lib/nls/loc/iconv** ディレクトリーにある fold7 コンバーターについて説明します。

コンバーター	説明
fold7_IBM-850	交換形式から IBM-850
fold7_IBM-921	交換形式から IBM-921
fold7_IBM-922	交換形式から IBM-922
fold7_IBM-932	交換形式から IBM-932
fold7_IBM-943	交換形式から IBM-943
fold7_IBM_1124	交換形式から IBM-1124
fold7_IBM_1129	交換形式から IBM-1129
fold7_IBM_eucCN	交換形式から IBM-eucCN
fold7_IBM-eucJP	交換形式から IBM-eucJP
fold7_IBM-eucKR	交換形式から IBM-eucKR
fold7_IBM-eucTW	交換形式から IBM-eucTW
fold7_ISO8859-1	交換形式から ISO8859-1
fold7_ISO8859-2	交換形式から ISO8859-2
fold7_ISO8859-3	交換形式から ISO8859-3
fold7_ISO8859-4	交換形式から ISO8859-4
fold7_ISO8859-5	交換形式から ISO8859-5
fold7_ISO8859-6	交換形式から ISO8859-6
fold7_ISO8859-7	交換形式から ISO8859-7
fold7_ISO8859-8	交換形式から ISO8859-8
fold7_ISO8859-9	交換形式から ISO8859-9
fold7_TIS-620	交換形式から TIS-620
fold7_UTF-8	交換形式から UTF-8
fold7_big5	交換形式から big5
fold7_GBK	交換形式から GBK
IBM-921_fold7	IBM-921 から交換形式
IBM-922_fold7	IBM-922 から交換形式
IBM-850_fold7	IBM-850 から交換形式
IBM-932_fold7	IBM-932 から交換形式
IBM-943_fold7	IBM-943 から交換形式
IBM-1124_fold7	IBM-1124 から交換形式
IBM-1129_fold7	IBM-1129 から交換形式
IBM-eucCN_fold7	IBM-eucCN から交換形式
IBM-eucJP_fold7	IBM-eucJP から交換形式
IBM-eucKR_fold7	IBM-eucKR から交換形式
IBM-eucTW_fold7	IBM-eucTW から交換形式
ISO8859-1_fold7	ISO8859-1 から交換形式
ISO8859-2_fold7	ISO8859-2 から交換形式

コンバーター	説明
ISO8859-3_fold7	ISO8859-3 から交換形式
ISO8859-4_fold7	ISO8859-4 から交換形式
ISO8859-5_fold7	ISO8859-5 から交換形式
ISO8859-6_fold7	ISO8859-6 から交換形式
ISO8859-7_fold7	ISO8859-7 から交換形式
ISO8859-8_fold7	ISO8859-8 から交換形式
ISO8859-9_fold7	ISO8859-9 から交換形式
TIS-620_fold7	TIS-620 から交換形式
UTF-8_fold7	UTF-8 から交換形式
big5_fold7	big5 から交換形式
GBK_fold7	GBK から交換形式

交換コンバーター - 8 ビット

このコンバーターでは、内部コードと 8 ビットの標準交換形式 (fold8) の間の変換が実現します。

fold8 名は、8 ビット・メール・プロトコルを使用してテキスト・データを渡すのに使用できるエンコードを示します。 これらのエンコードは、ISO2022 に基づいています。

fold8 コンバーターは、特定のコード・セット・エンコードからの文字を、それぞれの文字を示す標準 8 ビット・エンコードに変換します。このタイプの変換は、クライアントが、別のコード・セットと通信しているが、同じ文字セットを使用するようなネットワークで役立ちます。 例えば、以下のように使います。

コード・セット変換	説明
IBM-850 <—> ISO8859-1	共通ラテン語文字
IBM-932 <—> IBM-eucJP	共通日本語文字

以下のエスケープ・シーケンスは、標準のコード・セットを指定します。

エスケープ・シーケンス	標準コード・セット
01/11 02/04 02/09 04/01	GB2312.1980-0 の GR 右半分。
01/11 02/13 04/01	ISO8859-1 の GR 右半分。
01/11 02/13 04/02	ISO8859-2 の GR 右半分。
01/11 02/13 04/03	ISO8859-3 の GR 右半分。
01/11 02/13 04/04	ISO8859-4 の GR 右半分。
01/11 02/13 04/06	ISO8859-7 の GR 右半分。
01/11 02/13 04/07	ISO8859-6 の GR 右半分。
01/11 02/13 04/08	ISO8859-8 の GR 右半分。
01/11 02/13 04/13	ISO8859-5 の GR 右半分。
01/11 02/13 04/13	ISO8859-9 の GR 右半分。
01/11 02/09 04/09	JIS X0201.1976-1 の GR 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 JIS X0208.1978-1。
01/11 02/09 04/02	GR 7 ビット ASCII または ISO8859-1 の左半分。
01/11 02/05 02/15 03/01 M L 04/09 04/02 04/13 02/13 03/08 03/05 03/00 00/02	IBM-850 固有文字の GR 右半分。 ISO8859-1 と共通の文字では、このエスケープ・シーケンスは使用しません。

エスケープ・シーケンス	標準コード・セット
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	日本語ユーザー定義可能文字の GR 右半分。
01/11 02/08 04/02	GL 7 ビット ASCII または ISO8859-1 の左半分。
01/11 02/14 04/01	ISO8859-1 の GL 右半分。
01/11 02/14 04/02	ISO8859-2 の GL 右半分。
01/11 02/14 04/03	ISO8859-3 の GL 右半分。
01/11 02/14 04/04	ISO8859-4 の GL 右半分。
01/11 02/14 04/06	ISO8859-7 の GL 右半分。
01/11 02/14 04/07	ISO8859-6 の GL 右半分。
01/11 02/14 04/08	ISO8859-8 の GL 右半分。
01/11 02/14 04/12	ISO8859-5 の GL 右半分。
01/11 02/14 04/13	ISO8859-9 の GL 右半分。
01/11 02/08 04/09	JIS X0201.1976-0 の GL 右半分。
01/11 02/08 04/10	JIS X0201.1976 の GL 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 03/01 M L 06/09 06/02 06/13 02/13 03/08 03/05 03/00 00/02	IBM-850 固有文字の GL 右半分。 ISO8859-1 と共通の文字では、このエスケープ・シーケンスは使用しません。
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 日本語 (IBM-udcJP) ユーザー定義可能文字。
01/11 02/04 02/09 04/03	GR KSC5601-1987。
01/11 02/04 02/09 03/00	GR CNS11643-1986-1。
01/11 02/04 02/10 03/01	GR CNS11643-1986-2。
01/11 02/05 02/15 03/02 M L 04/09 04/02 04/13 02/13 07/05 06/04 06/03 05/05 05/08 00/02	中国語 (繁体字) ユーザー定義可能文字の GR 右半分。
01/11 02/05 02/15 03/02 M L 04/09 04/02 04/13 02/13 07/03 06/02 06/04 05/05 05/08 00/02	IBM-850 固有記号の GR 右半分。
01/11 02/04 02/08 04/03	GL KSC5601-1987。
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 05/05 05/08 00/02	GL 中国語 (繁体字) (IBM-udcTW) ユーザー定義可能文字。
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/03 06/02 06/04 05/05 05/08 00/02	GL 中国語 (繁体字) IBM-850 固有記号 (IBM-shdTW) ユーザー定義可能文字。
01/11 02/05 02/15 03/00 M L 05/05 05/04 04/06 02/13 03/08 00/02	UTF-8 としてエンコードされた UCS-2。上にリストされたいずれかのエスケープ・シーケンスでエンコードされていない文字の場合にのみ使用されます。

あるコード・セットから fold8 に変換する場合、コード・セットを指定するのに使用されるエスケープ・シーケンスは、リストされた順序で選択されます。例えば、JISX0208.1983-0 文字は、**01/11 02/04 02/08 04/02** を指定します。

関連概念:

87 ページの『libiconv の理解』
このセクションでは、**iconv** アプリケーション・プログラミング・インターフェース (API) による変換について記載します。

94 ページの『PC、ISO、および EBCDIC のコード・セット・コンバーター』
これらのコンバーターでは、PC、ISO、および EBCDIC 単一バイト・ステートレス・コード・セット間での変換を実現します。 次のタイプの変換がサポートされています。PC と ISO 間、PC と EBCDIC 間、

および ISO と EBCDIC 間。

ファイル

このリストでは、**/usr/lib/nls/loc/iconv** ディレクトリーにある fold8 コンバーターについて説明しています。

コンバーター	説明
fold8_IBM-850	交換形式から IBM-850
fold8_IBM-921	交換形式から IBM-921
fold8_IBM-922	交換形式から IBM-922
fold8_IBM-932	交換形式から IBM-932
fold8_IBM-943	交換形式から IBM-943
fold8_IBM-1124	交換形式から IBM-1124
fold8_IBM-1129	交換形式から IBM-1129
fold8_IBM-eucCN	交換形式から IBM-eucCN
fold8_IBM-eucJP	交換形式から IBM-eucJP
fold8_IBM-eucKR	交換形式から IBM-eucKR
fold8_IBM-eucTW	交換形式から IBM-eucTW
fold8_IBM-eucCN	交換形式から IBM-eucCN
fold8_ISO8859-1	交換形式から ISO8859-1
fold8_ISO8859-2	交換形式から ISO8859-2
fold8_ISO8859-3	交換形式から ISO8859-3
fold8_ISO8859-4	交換形式から ISO8859-4
fold8_ISO8859-5	交換形式から ISO8859-5
fold8_ISO8859-6	交換形式から ISO8859-6
fold8_ISO8859-7	交換形式から ISO8859-7
fold8_ISO8859-8	交換形式から ISO8859-8
fold8_ISO8859-9	交換形式から ISO8859-9
fold8_TIS-620	交換形式から TIS-620
fold8_UTF-8	交換形式から UTF-8
fold8_big5	交換形式から big5
fold8_GBK	交換形式から GBK
IBM-921_fold8	IBM-921 から交換形式
IBM-922_fold8	IBM-922 から交換形式
IBM-850_fold8	IBM-850 から交換形式
IBM-932_fold8	IBM-932 から交換形式
IBM-943_fold8	IBM-943 から交換形式
IBM-1124_fold8	IBM-1124 から交換形式
IBM-1129_fold8	IBM-1129 から交換形式
IBM-eucCN_fold8	IBM-eucCN から交換形式
IBM-eucJP_fold8	IBM-eucJP から交換形式
IBM-eucKR_fold8	IBM-eucKR から交換形式
IBM-eucTW_fold8	IBM-eucTW から交換形式
IBM-eucCN_fold8	IBM-eucCN から交換形式
ISO8859-1_fold8	ISO8859-1 から交換形式
ISO8859-2_fold8	ISO8859-2 から交換形式
ISO8859-3_fold8	ISO8859-3 から交換形式

コンバーター	説明
ISO8859-4_fold8	ISO8859-4 から交換形式
ISO8859-5_fold8	ISO8859-5 から交換形式
ISO8859-6_fold8	ISO8859-6 から交換形式
ISO8859-7_fold8	ISO8859-7 から交換形式
ISO8859-8_fold8	ISO8859-8 から交換形式
ISO8859-9_fold8	ISO8859-9 から交換形式
TIS-620_fold8	TIS-620 から交換形式
UTF-8_fold8	UTF-8 から交換形式
big5_fold8	big5 から交換形式
GBK_fold8	GBK から交換形式

交換コンバーター - 複合テキスト

複合テキスト交換コンバーターは、複合テキストと内部コード・セットとの間で変換します。

複合テキストは、X Consortium によって定義された交換エンコードです。これは、X クライアント間でテキストをやり取りするために使用されます。 複合テキストは、ISO2022 に基づいており、標準エスケープ・シーケンスを使用して、ほとんどの文字セットをエンコードできます。 さらに、専用文字セットをエンコードするための拡張機能も備えています。 サポートされているコード・セットでは、複合テキストに対するコンバーターが用意されています。 複合テキスト・エンコードを識別するのに使用される名前は、ct です。

標準コード・セットを指定するために、次のエスケープ・シーケンスがリストされている順序で使用されます。

01/11 02/05 02/15 03/01 M L 04/09 04/02 04/13 02/13 03/08 03/05 03/00 00/02
IBM-850 固有文字の GR 右半分。 ISO8859-1 と共通の文字では、このエスケープ・シーケンスは使用しません。

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
日本語ユーザー定義可能文字の GR 右半分。

01/11 02/05 02/15 03/01 M L 06/09 06/02 06/13 02/13 03/08 03/05 03/00 00/02
IBM-850 固有文字の GL 右半分。 ISO8859-1 と共通の文字では、このエスケープ・シーケンスは使用しません。

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 日本語 (IBM-udcJP) ユーザー定義可能文字。

ファイル

このリストでは、**/usr/lib/nls/loc/iconv** ディレクトリーにある複合テキスト・コンバーターについて説明しています。

コンバーター	説明
ct_IBM-850	交換形式から IBM-850
ct_IBM-921	交換形式から IBM-921
ct_IBM-922	交換形式から IBM-922
ct_IBM-932	交換形式から IBM-932
ct_IBM-943	交換形式から IBM-943
ct_IBM-1124	交換形式から IBM-1124
ct_IBM-1129	交換形式から IBM-1129
ct_IBM-eucCN	交換形式から IBM-eucCN
ct_IBM-eucJP	交換形式から IBM-eucJP
ct_IBM-eucKR	交換形式から IBM-eucKR
ct_IBM-eucTW	交換形式から IBM-eucTW
ct_ISO8859-1	交換形式から ISO8859-1
ct_ISO8859-2	交換形式から ISO8859-2
ct_ISO8859-3	交換形式から ISO8859-3
ct_ISO8859-4	交換形式から ISO8859-4
ct_ISO8859-5	交換形式から ISO8859-5
ct_ISO8859-6	交換形式から ISO8859-6
ct_ISO8859-7	交換形式から ISO8859-7
ct_ISO8859-8	交換形式から ISO8859-8
ct_ISO8859-9	交換形式から ISO8859-9
ct_TIS-620	交換形式から TIS-620
ct_big5	交換形式から big5
ct_GBK	交換形式から GBK
IBM-850_ct	IBM-850 から交換形式
IBM-921_ct	IBM-921 から交換形式
IBM-922_ct	IBM-922 から交換形式
IBM-932_ct	IBM-932 から交換形式
IBM-943_ct	IBM-943 から交換形式
IBM-1124_ct	IBM-1124 から交換形式
IBM-1129_ct	IBM-1129 から交換形式
IBM-eucCN_ct	IBM-eucCN から交換形式
IBM-eucJP_ct	IBM-eucJP から交換形式
IBM-eucKR_ct	IBM-eucKR から交換形式
IBM-eucTW_ct	IBM-eucTW から交換形式
ISO8859-1_ct	ISO8859-1 から交換形式
ISO8859-2_ct	ISO8859-2 から交換形式
ISO8859-3_ct	ISO8859-3 から交換形式
ISO8859-4_ct	ISO8859-4 から交換形式
ISO8859-5_ct	ISO8859-5 から交換形式
ISO8859-6_ct	ISO8859-6 から交換形式
ISO8859-7_ct	ISO8859-7 から交換形式
ISO8859-8_ct	ISO8859-8 から交換形式
ISO8859-9_ct	ISO8859-9 から交換形式
TIS-620_ct	TIS-620 から交換形式
big5_ct	big5 から交換形式
GBK_ct	GBK から交換形式

交換コンバーター — uucode

このコンバーターは、 **uuencode** および **uudecode** コマンドと同じようにマップします。

uucode からの変換時には、一度に 62 バイト (レコードの末尾の改行文字を含む) が変換され、 *outbuf* で 45 バイトが生成されます。

関連概念:

87 ページの『libiconv の理解』
このセクションでは、**iconv** アプリケーション・プログラミング・インターフェース (API) による変換について記載します。

ファイル

このリストでは、**/usr/lib/nls/loc/iconv** ディレクトリーにある uucode コンバーターについて説明しています。

コンバーター	説明
IBM-850_uucode	IBM-850 から uucode
IBM-921_uucode	IBM-921 から uucode
IBM-922_uucode	IBM-922 から uucode
IBM-932_uucode	IBM-932 から uucode
IBM-943_uucode	IBM-943 から uucode
IBM-1124_uucode	IBM-1124 から uucode
IBM-1129_uucode	IBM-1129 から uucode
IBM-eucJP_uucode	IBM-eucJP から uucode
IBM-eucKR_uucode	IBM-eucKR から uucode
IBM-eucTW_uucode	IBM-eucTW から uucode
IBM-eucCN_uucode	IBM-eucCN から uucode
ISO8859-1_uucode	ISO8859-1 から uucode
ISO8859-2_uucode	ISO8859-2 から uucode
ISO8859-3_uucode	ISO8859-3 から uucode
ISO8859-4_uucode	ISO8859-4 から uucode
ISO8859-5_uucode	ISO8859-5 から uucode
ISO8859-6_uucode	ISO8859-6 から uucode
ISO8859-7_uucode	ISO8859-7 から uucode
ISO8859-8_uucode	ISO8859-8 から uucode
ISO8859-9_uucode	ISO8859-9 から uucode
TIS-620_uucode	TIS-620 から uucode
big5_uucode	big5 から uucode
GBK_uucode	GBK から uucode
uucode_IBM-850	uucode から IBM-850
uucode_IBM-921	uucode から IBM-921
uucode_IBM-922	uucode から IBM-922
uucode_IBM-932	uucode から IBM-932
uucode_IBM-943	uucode から IBM-943
uucode_IBM-1124	uucode から IBM-1124
uucode_IBM-1129	uucode から IBM-1129

コンバーター	説明
uucode_IBM-eucCN	uucode から IBM-eucCN
uucode_IBM-eucJP	uucode から IBM-eucJP
uucode_IBM-eucKR	uucode から IBM-eucKR
uucode_IBM-eucTW	uucode から IBM-eucTW
uucode_ISO8859-1	uucode から ISO8859-1
uucode_ISO8859-2	uucode から ISO8859-2
uucode_ISO8859-3	uucode から ISO8859-3
uucode_ISO8859-4	uucode から ISO8859-4
uucode_ISO8859-5	uucode から ISO8859-5
uucode_ISO8859-6	uucode から ISO8859-6
uucode_ISO8859-7	uucode から ISO8859-7
uucode_ISO8859-8	uucode から ISO8859-8
uucode_ISO8859-9	uucode から ISO8859-9
uucode_TIS-1124	uucode から TIS-1124
uucode_big5	uucode から big5
uucode_GBK	uucode から GBK

UCS-2 交換コンバーター

UCS-2 では、ユニバーサル 16 ビット・エンコードを使用します。 コード・セットごとの変換は、コード・セットと UCS-2 との間で両方向に行われます。

UCS-2 コンバーターは、**/usr/lib/nls/loc/uconvTable**、および **/usr/lib/nls/loc/uconv** ディレクトリーにあります。 **uconvdef** コマンドは、新しいコンバーターを生成するため、または既存の UCS-2 コンバーターをカスタマイズするために使用されます。

コンバーター	説明
ISO8859-1	UCS-2 <—> ISO Latin-1
ISO8859-2	UCS-2 <—> ISO Latin-2
ISO8859-3	UCS-2 <—> ISO Latin-3
ISO8859-4	UCS-2 <—> ISO バルト語
ISO8859-5	UCS-2 <—> ISO キリル文字
ISO8859-6	UCS-2 <—> ISO アラビア語
ISO8859-7	UCS-2 <—> ISO ギリシャ語
ISO8859-8	UCS-2 <—> ISO ヘブライ語
ISO8859-9	UCS-2 <—> ISO トルコ語
JISX0201.1976-0	UCS-2 <—> 日本語 JISX0201-0
JISX0208.1983-0	UCS-2 <—> 日本語 JISX0208-0
CNS11643.1986-1	UCS-2 <—> 中国語 CNS11643-1
CNS11643.1986-2	UCS-2 <—> 中国語 CNS11643-2
KSC5601.1987-0	UCS-2 <—> 韓国語 KSC5601-0
IBM-eucCN	UCS-2 <—> 中国語 (簡体字) EUC
IBM-udcCN	UCS-2 <—> 中国語 (簡体字) ユーザー定義文字
IBM-sbdCN	UCS-2 <—> 中国語 (簡体字) IBM 固有文字
GB2312.1980-0	UCS-2 <—> 中国語 (簡体字) GB
IBM-1381	UCS-2 <—> 中国語 (簡体字) PC データ・コード

コンバーター	説明
IBM-935	UCS-2 <—> 中国語 (簡体字) EBCDIC
IBM-936	UCS-2 <—> 中国語 (簡体字) PC5550
IBM-eucJP	UCS-2 <—> 日本語 EUC
IBM-eucKR	UCS-2 <—> 韓国語 EUC
IBM-eucTW	UCS-2 <—> 中国語 (繁体字) EUC
IBM-udcJP	UCS-2 <—> 日本語ユーザー定義文字
IBM-udcTW	UCS-2 <—> 中国語 (繁体字) ユーザー定義文字
IBM-sbdTW	UCS-2 <—> 中国語 (繁体字) IBM 固有文字
UTF-8	UCS-2 <—> UTF-8
IBM-437	UCS-2 <—> USA PC データ・コード
IBM-850	UCS-2 <—> Latin-1 PC データ・コード
IBM-852	UCS-2 <—> Latin-2 PC データ・コード
IBM-857	UCS-2 <—> トルコ語 PC データ・コード
IBM-860	UCS-2 <—> ポルトガル語 PC データ・コード
IBM-861	UCS-2 <—> アイスランド語 PC データ・コード
IBM-863	UCS-2 <—> フランス語系カナダ語 PC データ・コード
IBM-865	UCS-2 <—> 北欧ゲルマン系言語 PC データ・コード
IBM-869	UCS-2 <—> ギリシャ語 PC データ・コード
IBM-921	UCS-2 <—> バルト語マルチリンガル・データ・コード
IBM-922	UCS-2 <—> エストニア語データ・コード
IBM-932	UCS-2 <—> 日本語 PC データ・コード
IBM-943	UCS-2 <—> 日本語 PC データ・コード
IBM-934	UCS-2 <—> 韓国 PC データ・コード
IBM-936	UCS-2 <—> 中華人民共和国 PC データ・コード
IBM-938	UCS-2 <—> 台湾語 PC データ・コード
IBM-942	UCS-2 <—> 拡張日本語 PC データ・コード
IBM-944	UCS-2 <—> 韓国語 PC データ・コード
IBM-946	UCS-2 <—> 中華人民共和国 SAA データ・コード
IBM-948	UCS-2 <—> 中国語 (繁体字) PC データ・コード
IBM-1124	UCS-2 <—> ウクライナ語 PC データ・コード
IBM-1129	UCS-2 <—> ベトナム語 PC データ・コード
TIS-620	UCS-2 <—> タイ PC データ・コード
IBM-037	UCS-2 <—> 米国、カナダ EBCDIC
IBM-273	UCS-2 <—> ドイツ、オーストリア EBCDIC
IBM-277	UCS-2 <—> デンマーク、ノルウェー EBCDIC
IBM-278	UCS-2 <—> フィンランド、スウェーデン EBCDIC
IBM-280	UCS-2 <—> イタリア EBCDIC
IBM-284	UCS-2 <—> スペイン、ラテンアメリカ EBCDIC
IBM-285	UCS-2 <—> 英国 EBCDIC
IBM-297	UCS-2 <—> フランス EBCDIC
IBM-500	UCS-2 <—> 国際 EBCDIC
IBM-875	UCS-2 <—> ギリシャ語 EBCDIC
IBM-930	UCS-2 <—> 日本語カタカナ漢字 EBCDIC
IBM-933	UCS-2 <—> 韓国語 EBCDIC
IBM-937	UCS-2 <—> 中国語 (繁体字) EBCDIC
IBM-939	UCS-2 <—> 日本語英数小文字漢字 EBCDIC

コンバーター	説明
IBM-1026	UCS-2 <—> トルコ語 EBCDIC
IBM-1112	UCS-2 <—> バルト語マルチリンガル EBCDIC
IBM-1122	UCS-2 <—> エストニア語 EBCDIC
IBM-1124	UCS-2 <—> ウクライナ語 EBCDIC
IBM-1129	UCS-2 <—> ベトナム語 EBCDIC
TIS-620	UCS-2 <—> タイ EBCDIC

関連概念:

『UTF-8 交換コンバーター』
このセクションでは、各コード・セットと UTF-8 の両方向に行われる変換について説明します。

UTF-8 交換コンバーター

このセクションでは、各コード・セットと UTF-8 の両方向に行われる変換について説明します。

UTF-8 は、ユニバーサルなマルチバイト・エンコードです。コード・セットごとの変換は、コード・セットと UTF-8 との間で両方向に行われます。

UTF-8 変換は、通常は Universal_UCS_Conv および **/usr/lib/nls/loc/uconv/UTF-8** コンバーターを使用することによって行われます。

コンバーター	説明
ISO8859-1	UTF-8 <—> ISO Latin-1
ISO8859-2	UTF-8 <—> ISO Latin-2
ISO8859-3	UTF-8 <—> ISO Latin-3
ISO8859-4	UTF-8 <—> ISO バルト語
ISO8859-5	UTF-8 <—> ISO キリル文字
ISO8859-6	UTF-8 <—> ISO アラビア語
ISO8859-7	UTF-8 <—> ISO ギリシャ語
ISO8859-8	UTF-8 <—> ISO ヘブライ語
ISO8859-9	UTF-8 <—> ISO トルコ語
JISX0201.1976-0	UTF-8 <—> 日本語 JISX0201-0
JISX0208.1983-0	UTF-8 <—> 日本語 JISX0208-0
CNS11643.1986-1	UTF-8 <—> 中国語 CNS11643-1
CNS11643.1986-2	UTF-8 <—> 中国語 CNS11643-2
KSC5601.1987-0	UTF-8 <—> 韓国語 KSC5601-0
IBM-eucCN	UTF-8 <—> 中国語 (簡体字) EUC
IBM-eucJP	UTF-8 <—> 日本語 EUC
IBM-eucKR	UTF-8 <—> 韓国語 EUC
IBM-eucTW	UTF-8 <—> 中国語 (繁体字) EUC
IBM-udcJP	UTF-8 <—> 日本語ユーザー定義文字
IBM-udcTW	UTF-8 <—> 中国語 (繁体字) ユーザー定義文字
IBM-sbdTW	UTF-8 <—> 中国語 (繁体字) IBM 固有文字
UCS-2	UTF-8 <—> UCS-2
IBM-437	UTF-8 <—> USA PC データ・コード
IBM-850	UTF-8 <—> Latin-1 PC データ・コード
IBM-852	UTF-8 <—> Latin-2 PC データ・コード

コンバーター	説明
IBM-857	UTF-8 <—> トルコ語 PC データ・コード
IBM-860	UTF-8 <—> ポルトガル語 PC データ・コード
IBM-861	UTF-8 <—> アイスランド語 PC データ・コード
IBM-863	UTF-8 <—> フランス語系カナダ語 PC データ・コード
IBM-865	UTF-8 <—> 北欧ゲルマン系言語 PC データ・コード
IBM-868	UTF-8 <—> ウルドゥー語 IBM-868
IBM-869	UTF-8 <—> ギリシャ語 PC データ・コード
IBM-918	UTF-8 <—> ウルドゥー語 IBM-918
IBM-921	UTF-8 <—> バルト語マルチリンガル・データ・コード
IBM-922	UTF-8 <—> エストニア語データ・コード
IBM-932	UTF-8 <—> 日本語 PC データ・コード
IBM-943	UTF-8 <—> 日本語 PC データ・コード
IBM-934	UTF-8 <—> 韓国 PC データ・コード
IBM-935	UTF-8 <—> 中国語 (簡体字) EBCDIC
IBM-936	UTF-8 <—> 中華人民共和国 PC データ・コード
IBM-938	UTF-8 <—> 台湾語 PC データ・コード
IBM-942	UTF-8 <—> 拡張日本語 PC データ・コード
IBM-944	UTF-8 <—> 韓国語 PC データ・コード
IBM-946	UTF-8 <—> 中華人民共和国 SAA データ・コード
IBM-948	UTF-8 <—> 中国語 (繁体字) PC データ・コード
IBM-1006	UTF-8 <—> ウルドゥー語 IBM-1006
IBM-1124	UTF-8 <—> ウクライナ語 PC データ・コード
IBM-1129	UTF-8 <—> ベトナム語 PC データ・コード
TIS-620	UTF-8 <—> タイ PC データ・コード
IBM-037	UTF-8 <—> 米国、カナダ EBCDIC
IBM-273	UTF-8 <—> ドイツ、オーストリア EBCDIC
IBM-277	UTF-8 <—> デンマーク、ノルウェー EBCDIC
IBM-278	UTF-8 <—> フィンランド、スウェーデン EBCDIC
IBM-280	UTF-8 <—> イタリア EBCDIC
IBM-284	UTF-8 <—> スペイン、ラテンアメリカ EBCDIC
IBM-285	UTF-8 <—> 英国 EBCDIC
IBM-297	UTF-8 <—> フランス EBCDIC
IBM-500	UTF-8 <—> 国際 EBCDIC
IBM-875	UTF-8 <—> ギリシャ語 EBCDIC
IBM-930	UTF-8 <—> 日本語カタカナ漢字 EBCDIC
IBM-933	UTF-8 <—> 韓国語 EBCDIC
IBM-937	UTF-8 <—> 中国語 (繁体字) EBCDIC
IBM-939	UTF-8 <—> 日本語英数小文字漢字 EBCDIC
IBM-1026	UTF-8 <—> トルコ語 EBCDIC
IBM-1112	UTF-8 <—> バルト語マルチリンガル EBCDIC
IBM-1122	UTF-8 <—> エストニア語 EBCDIC
IBM-1124	UTF-8 <—> ウクライナ語 EBCDIC
IBM-1129	UTF-8 <—> ベトナム語 EBCDIC
IBM-1381	UTF-8 <—> 中国語 (簡体字) PC データ・コード
GB18030	UTF-8<—> 中国語 (簡体字)
TIS-620	UTF-8 <—> タイ EBCDIC

関連概念:

98 ページの『マルチバイト・コード・セット・コンバーター』
このセクションでは、マルチバイト・コード・セット・コンバーターが変換に使用するコード・セットについて説明します。

82 ページの『UCS-2 および UTF-8』
AIX には、特定言語または言語グループの必要を満たすコード・セット群が用意されています。 ISO8859 ファミリーのコード・セット、PC コード・セット、または拡張 UNIX コード (EUC) コード・セット内に表現されているコード・セットでは、異なるスクリプトの文字を混合させることはできません。
ISO8859-1 を使用すると、Latin 1 文字 (基本的に、米国、カナダ、西欧、およびラテンアメリカで話されている言語) を混合し表現できます。 ISO8859-2 は西ヨーロッパ言語を、ISO8859-5 はキリル文字を、ISO8859-6 はアラビア語を、 ISO8859-7 はギリシャ語を、ISO8859-8 はヘブライ語を、ISO8859-9 はトルコ語を、 IBM-eucJP は日本語を、IBM-eucKR は韓国語を、IBM-eucTW は中国語 (繁体字) を網羅します。 つまり、上記のコード・セットの中で、すべての言語が網羅されているものはないということです。

110 ページの『UCS-2 交換コンバーター』
UCS-2 では、ユニバーサル 16 ビット・エンコードを使用します。 コード・セットごとの変換は、コード・セットと UCS-2 との間で両方向に行われます。

各種コンバーター

コード・セットで使用される一群の低レベル・コンバーターおよび交換コンバーターが用意されています。これらのコンバーターのことを、その他のコンバーター と呼んでいます。 いくつかの交換コンバーターで、これらの低レベル・コンバーターを使用できます。 しかし、これらのコンバーターは、他のコンバーターのサポートを目的としているため、使用することはお勧めしません。

ファイル

このリストでは、**/usr/lib/nls/loc/iconv** ディレクトリーと **/usr/lib/nls/loc/iconvTable** ディレクトリーにある各種コンバーターについて説明しています。

コンバーター	説明
IBM-932_JISX0201.1976-0	IBM-932 から JISX0201.1976-0
IBM-932_JISX0208.1983-0	IBM-932 から JISX0208.1983-0
IBM-932_IBM-udcJP	IBM-932 から IBM-udcJP (日本語ユーザー定義文字)
IBM-943_JISX0201.1976-0	IBM-943 から JISX0201.1976-0
IBM-943_JISX0208.1983-0	IBM-943 から JISX0208.1983-0
IBM-943_IBM-udcJP	IBM-943 から IBM-udcJP (日本語ユーザー定義文字)
IBM-eucJP_JISX0201.1976-0	IBM-eucJP から JISX0201.1976-0
IBM-eucJP_JISX0208.1983-0	IBM-eucJP から JISX0208.1983-0
IBM-eucJP_IBM-udcJP	IBM-eucJP から IBM-udcJP (日本語ユーザー定義文字)
IBM-eucKR_KSC5601.1987-0	IBM-eucKR から KSC5601.1987-0
IBM-eucTW_CNS11643.1986-1	IBM-eucTW から CNS11643.1986.1
IBM-eucTW_CNS11643.1986-2	IBM-eucTW から CNS11643.1986-2
IBM-eucCN_GB2312.1980-0	IBM-eucCN から GB2312.1980-0

iconv インターフェースを使用したコンバーターの作成

このセクションでは、コード・セット・コンバーターを作成する準備として、 iconv サブルーチンおよび構造体についての情報を提供します。 この説明には、フレームワークを操作する制御フローおよび順序の概要、コード・セット・コンバーターを作成することの詳細、そして、コード、ヘッダー・ファイル、および makefile を含む例が含まれます。 このセクションは、AIX 内の **iconv** フレームワークに適用されます。

iconv_open、**iconv**、および **iconv_close** サブルーチンのフレームワークの下で、いくつかの異なるタイプのコンバーターを作成して使用できます。 アプリケーションは、これらのサブルーチンを呼び出して、あるコード・セットの文字を、別のコード・セットの文字に変換することができます。 **iconv_open**、**iconv**、および **iconv_close** サブルーチンへのアクセスと使用は、X/Open Portability Guide Issue 4 によって標準化されています。

コード・セットとコンバーター

コード・セットは、ステートフル・エンコードとステートレス・エンコードの 2 つのカテゴリーに分類できます。

ステートフル・コード・セットとコンバーター

ステートフル・エンコードでは、シフトイン・コードとシフトアウト・コードを使用して状態を変更します。 シフトアウトは、文字のデータ・ストリームでホスト 2 バイト・データが開始したことを示し、シフトインは、この 2 バイト文字データが終了したことを示すために使用できます。 2 バイト・データがなければ、1 バイト文字データの開始を通知します。 そのようなステートフル・コード・セットの一例は、主にメインフレーム (ホスト) で使用される IBM-930 です。

ステートフル・エンコードを別のコード・セットへ変換するよう作成されたコンバーターは、余分な処理が必要になるため、複雑になりがちです。

ステートレス・コード・セットとコンバーター

このセクションでは、ステートレス・コード・セットとコンバーターについて説明します。

ステートレス・コード・セットは、次のいずれかのタイプに分類できるものです。

- ISO8859 ファミリー (ISO8859-1、ISO8859-2、など) のような単一バイト・コード・セット
- IBM-eucJP (日本語)、IBM-932 (シフト JIS) のようなマルチバイト・コード・セット

コード・セットが同じ文字を表現している場合にのみ、変換は意味があります。

コード・セット変換の一番簡単なタイプは、ISO8859-1 から IBM-850 へのコンバーターのような、単一バイト・コード・セット・コンバーターにあります。これらの単一バイト・コード・セット・コンバーターは、簡単なテーブル・ベースの変換に基づいています。 IBM-eucJP から IBM-932 のような、マルチバイト文字エンコードの変換は、一般的に、テーブルではなくアルゴリズムに基づいています。これは、テーブルは冗長になりがちなためです。

iconv フレームワーク構造体の概要

iconv フレームワークは、**iconv_open**、**iconv**、および **iconv_close** サブルーチンで構成されており、すべてのコンバーターの一部である共通のコア構造体に基づいています。 コア構造体は、コンバーター・オブジェクト・モジュールのロード時に初期設定されます。 コンバーターのロードが完了したら、主要なエントリー・ポイント (必ず **instantiate** サブルーチン) が呼び出されます。 これにより、コア構造体が初期設定され、コア・コンバーター・ディスクリプターが戻されます。 これは、コンバーター固有の構造体を

割り当てるためにコンバーターによって提供される **init** サブルーチンを呼び出すときにも使用されます。この **init** サブルーチンは、コア・コンバーター・ディスクリプターへのポインターを持つ、別のコンバーター・ディスクリプターを戻します。 **init** サブルーチンは、必要に応じてメモリーを割り当て、必要であれば他のコンバーターを呼び出すことができます。 **init** サブルーチンは、コンバーター固有の初期設定のための場所であるのに対し、 **instantiate** サブルーチンは、汎用のエントリー・ポイントです。

このコンバーターのコンバーター・ディスクリプターが割り当てられて初期設定されたら、次のステップは、機能の **exec** 部分に必要な実際のコードを提供することです。 コンバーターがテーブル・ベースのコンバーターである場合、必要なことは、 **genxlt** ユーティリティーの入力上の必要に適したソース・ファイル形式を提供することだけです。これにより、このソース・テーブルが入力と見なされ、**iconv** フレームワークで使用できる出力ファイル形式が生成されます。

iconv.h ファイルと構造体

このセクションでは、iconv.h file ファイルとその構造体について説明します。

/usr/include の **iconv.h** ファイルでは、次の構造体が定義されています。

```
typedef struct __iconv_rec   iconv_rec, *iconv_t;
struct __iconv_rec   {
    _LC_object_t  hdr;
    iconv_t (*open)(const char *tocode, const char *fromcode);
    size_t (*exec)(iconv_t cd, char **inbuf, size_t *inbytesleft,
            char **outbuf, size_t *outbytesleft);
    void  (*close)(iconv_t cd);
};
```

一般的なコア構造体は、次のとおりです (**/usr/include/iconv.h**)。

```
typedef struct _LC_core_iconv_type      _LC_core_iconv_t;
struct _LC_core_iconv_type {
        _LC_object_t  hdr;
        /* implementation initialization */
        _LC_core_iconv_t        *(*init)();
        size_t  (*exec)();
        void    (*close)();
};
```

各コンバーターには静的メモリー領域があり、そこに **_LC_core_iconv_t** 構造体があります。 これは、コンバーター・プログラムの一部として提供されている **instantiate** サブルーチンで初期設定されます。

iconv 制御フロー

アプリケーションは、次の呼び出しでコード・セット・コンバーターを呼び出します。

```
iconv_open(char *to_codeset, char *from_codeset)
```

to および *from* コード・セットは、 **LOCPATH** 環境変数で定義された検索パスを使用してコンバーターを選択するときに使用されます。 **iconv_open** サブルーチンは、**_lc_load** サブルーチンを使用して、 *from* および *to* コード・セット名を **iconv_open** サブルーチンに連結することで指定したオブジェクト・モジュールをロードします。

```
CONVERTER NAME= "from_codeset" + "_" +"to_codeset"
```

from_codeset が IBM-850 で、*to_codeset* が ISO8859-1 の場合、コンバーター名は IBM-850_ISO8859-1 になります。

コンバーターをロードしたら、**_lc_load** ローダー・サブルーチンにより、エントリー・ポイントが呼び出されます。 これは、コンバーターに対する最初の呼び出しになります。 次に instantiate サブルーチン

は、 **_LC_core_iconv_t** コア構造体を初期設定します。 それから **iconv_open** サブルーチンは、戻されたコア構造体に関連付けられた **init** サブルーチンを呼び出します。 init サブルーチンは、コンバーター固有のディスクリプター構造体を割り振り、必要に応じてコンバーターによって初期設定します。 **iconv_open** サブルーチンは、このコンバーター固有の構造体を戻します。 しかし、戻り値は、ユーザーのアプリケーションでは、 **iconv_t** への型キャストになります。 したがって、アプリケーション側からは、コンバーター固有の構造体全体は見えません。公開されている **iconv_t** 構造体だけが見えます。 コンバーター・コード自体は、専用コンバーター構造体を使用しています。 **iconv** コンバーターを使用するアプリケーションでは、コンバーター・ディスクリプターを変更しないようにします。コンバーター・ディスクリプターは、隠された構造体として使用するようにします。

エントリー・ポイント は、コンバーターが *iconv_open* サブルーチンへの呼び出しでオープンするときに、そのエントリー・ポイントが自動的に呼び出されるように、それぞれのコンバーターで宣言されます。このエントリー・ポイントは、すべてのコンバーターに付属していなければならない *instantiate* サブルーチンです。 エントリー・ポイントは、makefile の中で次のように指定されます。

```
LDENTRY=-einstantiate
```

iconv_open サブルーチンへの呼び出しでコンバーターがロードされると、 **instantiate** サブルーチンが呼び出されます。 このサブルーチンは、静的なコア変換ディスクリプター構造体の **_LC_core_iconv_t cd** を初期設定します。

コア変換ディスクリプター **cd** には、特定のコンバーターによって提供された **init**、**_iconv_exec**、および _iconv_close の各サブルーチンへのポインターが含まれます。 **instantiate** サブルーチンは、後で使用するコア変換ディスクリプターを戻します。 _LC_core_iconv_t 構造体は、**/usr/include/iconv.h** 内で定義されます。

iconv_open サブルーチンが呼び出されると、次のアクションが起こります。

1. **LOCPATH** 環境変数を使用してコンバーターが見つけられてロードされ、 **instantiate** サブルーチンが呼び出されます。 成功したら、コア変換ディスクリプターが戻されます。 (**_LC_core_iconv_t *cd**)。 コンバーターに付属する **instantiate** サブルーチンは、コア構造体のヘッダーの初期設定を担当します。
2. 次に **iconv_open** サブルーチンは、コア変換ディスクリプターで指定された **init** サブルーチンを呼び出します。 コンバーターに備えられている **init** サブルーチンは、このコンバーターに必要なコンバーター・ディスクリプターを保存するのに要するメモリーの割り当てを担当します。例えば、ステートレス・コンバーターで必要な構造体は次のようになります。

```
typedef struct _LC_sample_iconv_rec {
                  LC_core_iconv_t        core;

            } _LC_sample_iconv_t;

To initialize this, the converter has to do the following in the
init subroutine:

static _LC_sample_iconv_t*
init (_LC_core_iconv_t *core_cd, char* toname,char* fromname)
{
   _LC_sample_iconv_t     *cd;    /* converter descriptor */

    /*
    **        Allocate a converter descriptor
    **/
    if(!(cd = ( _LC_sample_iconv_t *) malloc (
                    sizeof(_LC_sample_iconv_t ))))
            return (NULL);
```

```
        /*
        ** Copy the core part of converter descriptor which is
        ** passed in
        */
        cd-&gt;core = *core_cd;
        /*
        **      Return the converter descriptor
        */
        return cd;
    }
```

アプリケーションは **iconv** サブルーチンを呼び出し、実際のコード・セット変換を実行します。 **iconv** サブルーチンは、コア構造体の **exec** サブルーチンを呼び出します。

アプリケーションは、**iconv_close** サブルーチンを呼び出し、変換のために割り当てられていたすべてのメモリーを解放します。 **iconv_close** サブルーチンは、コア構造体の **close** サブルーチンを呼び出します。

コード・セット・コンバーターの作成

このセクションでは、すでに説明した概念を使用してコンバーターを作成する方法について説明します。

各コンバーターは、次のサブルーチンを定義する必要があります。

- **instantiate**
- **init**
- **iconv_exec**
- **iconv_close**

コンバーター固有の構造体には、最初の要素としてコア **iconv** 構造体が必要です。 例えば、次のようにします。

```
        typedef struct _LC_example_rec {
            /* Core should be the first element */
                _LC_core_iconv_t        core;
            /* The rest are converter specific data (optional) */
                iconv_t         curcd;
                iconv_t         sb_cd;
                iconv_t         db_cd;
                unsigned char   *cntl;
        } _LC_example_iconv_t;
```

別のコンバーター構造体は次のとおりです。

```
        typedef struct _LC_sample_iconv_rec {
                _LC_core_iconv_t        core;
        } _LC_sample_iconv_t;
```

アルゴリズム・ベースのステートレス・コンバーター

各コンバーターには、以前に指定したサブルーチンが必要です。 詳細のないサブルーチン・ヘッダーだけが提供されますが、すべてのコンバーターに共通の **instantiate** サブルーチンは例外で、同じ方法でコード化されていなければなりません。

次に示すアルゴリズム・ベースのステートレス・コンバーターの例は、 IBM-850 コード・セットから ISO8859-1 コード・セットへのサンプル・コンバーターです。

```
#include <stdlib.h>
#include <iconv.h>
#include "850_88591.h"
/*
 *      Name :  _iconv_exec()
```

```
 *
 *      This contains actual conversion method.
 */
static size_t   _iconv_exec(_LC_sample_iconv_t *cd,
                        unsigned char** inbuf,
                        size_t *inbytesleft,
                        unsigned char** outbuf,
                        size_t *outbytesleft)
/*
 *      cd              : converter descriptor
 *      inbuf           : input buffer
 *      outbuf          : output buffer
 *      inbytesleft     : number of data(in bytes) in input buffer
 *      outbytesleft    : number of data(in bytes) in output buffer
 */

{
}

/*
 *      Name :   _iconv_close()
 *
 *      Free the allocated converter descriptor
 */
static void      _iconv_close(iconv_t cd)
{
}

/*
 *      Name :  init()
 *
 *      This allocates and initializes the converter descriptor.
 */
static _LC_sample_iconv_t        *init (_LC_core_iconv_t *core_cd,
                                  char* toname, char* fromname)
{
}

/*
 *       Name :  instantiate()
 *
 *       Core part of a converter descriptor is initialized here.
 */
_LC_core_iconv_t         *instantiate(void)
{
        static _LC_core_iconv_t   cd;

        /*
        * * Initialize _LC_MAGIC and _LC_VERSION are
        ** defined in <lc_core.h>. _LC_ICONV and _LC_core_iconv_t
        ** are defined in <iconv.h>.
         */
        cd.hdr.magic = _LC_MAGIC;
        cd.hdr.version = _LC_VERSION;
        cd.hdr.type_id = _LC_ICONV;
        cd.hdr.size = sizeof (_LC_core_iconv_t);

        /*
         *      Set pointers to each method.
         */
        cd.init = init;
        cd.exec = _iconv_exec;
        cd.close = _iconv_close;

        /*
```

```
         *      Returns the core part
         */
        return &cd;
}
```

ステートフル・コンバーター

ステートフル・コンバーターにはさらに多くの情報が必要なため、コンバーターに関係したその他の情報が提供されています。 次に示すステートフル・コンバーターの例は、 IBM-930 コード・セットから IBM-932 コード・セットへのサンプル・コンバーターです。

host.h ファイルには、次の構造体が入っています。

```
typedef struct _LC_host_iconv_rec {
                _LC_core_iconv_t        core;
                iconv_t         curcd;
                iconv_t         sb_cd;
                iconv_t         db_cd;
                unsigned char   *cntl;
        } _LC_host_iconv_t;

#include <stdlib.h&gt;
#include <sys/types.h&gt;
#include <iconv.h&gt;
#include "host.h"

/*
** The _iconv_exec subroutine to be invoked via cd-&gt;exec()
*/
static int      _iconv_exec(_LC_host_iconv_t *cd,
        unsigned char **inbuf, size_t *inbytesleft,
        unsigned char **outbuf, size_t *outbytesleft)
{
        unsigned char    *in, *out;
        int              ret_value;

    if (!cd){
        errno = EBADF; return NULL;
    }

    if (!inbuf) {
        cd-&gt;curcd = cd-&gt;sb_cd;
        return ICONV_DONE;
    }

    do {
    if ((ret_value = iconv(cd-&gt;curcd, inbuf, inbytesleft, outbuf,
             outbytesleft)) != ICONV_INVAL)
             return ret_value;
        in = *inbuf;
        out = *outbuf;
        if (in[0] == SO) {
            if (cd-&gt;curcd == cd-&gt;db_cd){
                errno = EILSEQ;
                return ICONV_INVAL;
            }
            cd-&gt;curcd = cd-&gt;db_cd;
        }
        else if (in[0] == SI) {
            if (cd-&gt;curcd == cd-&gt;sb_cd){
                errno = EILSEQ;
                return ICONV_INVAL;
            }
                cd-&gt;curcd = cd-&gt;sb_cd;
        }else if (in[0] <= 0x3f &&
            cd-&gt;curcd == cd-&gt;sb_cd) {
```

```
            if (*outbytesleft < 1){
                errno = E2BIG;
                return ICONV_OVER;
            }
                out[0] = cd-&gt;cntl[in[0]];
                *outbuf = ++out;
                (*outbytesleft)--;
        }
        else {
            errno = EILSEQ; return ICONV_INVAL;
        }
        *inbuf = ++in;
        (*inbytesleft)--;
    } while (1);
}

/*
** The iconv_close subroutine is a macro accessing this
** subroutine as set in the core iconv structure.
*/
static void   _iconv_close(_LC_host_iconv_t *cd)
{
    if (cd) {
        if (cd-&gt;sb_cd)
            iconv_close(cd-&gt;sb_cd);
        if (cd-&gt;db_cd)
            iconv_close(cd-&gt;db_cd);
        free(cd);
    }else{
        errno = EBADF;
    }
}

/*
** The init subroutine to be invoked when iconv_open() is called.
*/
static _LC_host_iconv_t   *init(_LC_core_iconv_t *core_cd,
               char* toname, char* fromname)
{
        _LC_host_iconv_t*  cd;
        int   i;

    for (i = 0; 1; i++) {
        if (!_iconv_host[i].local)
            return NULL;
        if (strcmp(toname, _iconv_host[i].local) == 0 &&
            strcmp(fromname, _iconv_host[i].host) == 0)
            break;
    }

        if (!(cd = (_LC_host_iconv_t *)
                    malloc(sizeof(_LC_host_iconv_t))))
                return (NULL);

    if (!(cd-&gt;sb_cd = iconv_open(toname, _iconv_host[i].sbcs))) {
        free(cd);
        return NULL;
    }
    if (!(cd-&gt;db_cd = iconv_open(toname, _iconv_host[i].dbcs))) {
        iconv_close(cd-&gt;sb_cd);
        free(cd);
        return NULL;
    }
            cd-&gt;core = *core_cd;
    cd-&gt;cntl = _iconv_host[i].fcntl;
    cd-&gt;curcd = cd-&gt;sb_cd;
    return cd;
```

```
}

/*
** The instantiate() method is called when iconv_open() loads the
** converter by a call to __lc_load().
*/
_LC_core_iconv_t         *instantiate(void)
{
        static _LC_core_iconv_t
cd;

        cd.hdr.magic = _LC_MAGIC;
        cd.hdr.version = _LC_VERSION;
        cd.hdr.type_id = _LC_ICONV;
        cd.hdr.size = sizeof (_LC_core_iconv_t);
        cd.init = init;
        cd.exec = _iconv_exec;
        cd.close = _iconv_close;
        return &cd;
}
```

例

この例では、IBM-850 コード・セットから ISO8859-1 コード・セットへのアルゴリズム・ベースの変換を実行する、ステートレス・コンバーターのサンプル・コードが提供されます。

この例のファイル名は、850_88591.c です。

```
#include <stdlib.h>
#include <iconv.h>
#include "850_88591.h"

#define DONE     0

/*
 *  Name :  _iconv_exec()
 *
 *  This contains actual conversion method.
 */
static size_t  _iconv_exec(_LC_sample_iconv_t *cd,
      unsigned char** inbuf, size_t *inbytesleft,
      unsigned char** outbuf, size_t *outbytesleft)
/*
 *  cd      : converter descriptor
 *  inbuf       : input buffer
 *  outbuf      : output buffer
 *  inbytesleft   : number of data(in bytes) in input buffer
 *  outbytesleft    : number of data(in bytes) in output buffer
 */
{
        unsigned char   *in;   /* point the input buffer */
        unsigned char   *out;  /* point the output buffer */
        unsigned char   *e_in; /* point the end of input buffer*/
        unsigned char   *e_out; /* point the end of output buffer*/

    /*
     *   If given converter discripter is invalid,
     *   it sets the errno and returns the number
     *   of bytes left to be converted.
     */
        if (!cd) {
        errno = EBADF;
        return *inbytesleft;
        }

        /*
```

```
         *      If the input buffer does not exist or there
         *      is no character to be converted, it returns
         *      0 (no characters to be converted).
         */
        if (!inbuf || !(*inbytesleft))
                return DONE;

        /*
         *      Set up pointers and initialize other variables
         */
        e_in = (in = *inbuf) + *inbytesleft;
        e_out = (out = *outbuf) + *outbytesleft;

     /*
      *     Perform code point conversion until all input
      *     is consumed.
      *     When error occurs (i.e. buffer overflow), error
      *     number is set and exit this loop.
      */
         while (in < e_in) {

        /*
         *   If there is not enough space left in output buffer
         *   to hold the converted data, it stops converting and
         *   sets the errno to E2BIG.
         */
        if (e_out <= out) {
            errno = E2BIG;
            break;
        }

        /*
         *   Convert the input data and store it into the output
         *   buffer, then advance the pointers which point to the
         *   buffers.
         */
        *out++ = table[*in++];

    }   /* while */

    /*
     *  Update the pointers to the buffers and
     *  input /output byte counts
     */
     *inbuf = in;
        *outbuf = out;
    *inbytesleft = e_in - in;
    *outbytesleft = e_out - out;

    /*
     *   Reurn the number of bytes left to be converted
     *   (0 for successful conversion completion)
     */
    return *inbytesleft;
}

/*
 *  Name :  _iconv_close()
 *
 *  Free the allocated converter descriptor
 */
static void _iconv_close(iconv_t cd)
{
    if (!cd)
        free(cd);
    else
        /*
```

```
         *  If given converter is not valid,
         *  it sets the errno to EBADF
         */
        errno = EBADF;
}

/*
 *  Name :  init()
 *
 *  This allocates and initializes the converter descriptor.
 */
static _LC_sample_iconv_t*
init (_LC_core_iconv_t *core_cd,  char* toname, char* fromname)
{
        _LC_sample_iconv_t *cd; /* converter descriptor */

    /*
     *   Allocate a converter descriptor
     */
        if (!(cd = (_LC_sample_iconv_t *)
                        malloc(sizeof(_LC_sample_iconv_t))))
                return (NULL);

    /*
     *Copy the core part of converter descriptor which is passed       *in
     */
        cd->core = *core_cd;

    /*
     *   Return the converter descriptor
     */
        return cd;
}

/*
 *  Name :  instantiate()
 *
 *  Core part of a converter descriptor is initialized here.
 */
_LC_core_iconv_t*   instantiate(void)
{
        static _LC_core_iconv_t  cd;

    /*
     *  Initialize
     *  _LC_MAGIC and _LC_VERSION are defined in <lc_core.h>.
     *  _LC_ICONV and _LC_core_iconv_t are defined in <iconv.h>.
     */
    cd.hdr.magic = _LC_MAGIC;
    cd.hdr.version = _LC_VERSION;
    cd.hdr.type_id = _LC_ICONV;
    cd.hdr.size = sizeof (_LC_core_iconv_t);

    /*
     *  Set pointers to each method.
     */
        cd.init = init;
        cd.exec = _iconv_exec;
        cd.close = _iconv_close;

    /*
     *  Returns the core part
     */
        return &cd;
}
```

この例には、850_88591.h というサンプル・ヘッダー・ファイルが含まれます。

```
#ifndef _ICONV_SAMPLE_H
#define _ICONV_SAMPLE_H

/*
 *      Define _LC_sample_iconv_t
 */
typedef struct _LC_sample_iconv_rec {
        _LC_core_iconv_t   core;
} _LC_sample_iconv_t;

static unsigned char     table[] = { /*


        ┌──────────────────────────────────────┐
        │      IBM-850              ISO8859-1  │
        └──────────────────────────────────────┘
        /*      0x00     */              0x00,
        /*      0x01     */              0x01,
        /*      0x02     */              0x02,
        /*      0x03     */              0x03,
        /*      0x04     */              0x04,
        /*      0x05     */              0x05,
        /*      0x06     */              0x06,
        /*      0x07     */              0x07,
        /*      0x08     */              0x08,
        /*      0x09     */              0x09,
        /*      0x0A     */              0x0A,
        /*      0x0B     */              0x0B,
        /*      0x0C     */              0x0C,
        /*      0x0D     */              0x0D,
        .
        .
        .
        /*      0xF3     */              0xBE,
        /*      0xF4     */              0xB6,
        /*      0xF5     */              0xA7,
        /*      0xF6     */              0xF7,
        /*      0xF7     */              0xB8,
        /*      0xF8     */              0xB0,
        /*      0xF9     */              0xA8,
        /*      0xFA     */              0xB7,
        /*      0xFB     */              0xB9,
        /*      0xFC     */              0xB3,
        /*      0xFD     */              0xB2,
        /*      0xFE     */              0x1A,
        /*      0xFF     */              0xA0,
};
#endif
```

この例は、サンプル makefile です。

```
SHELL   = /bin/ksh
CFLAGS  = $(COMPOPT) $(INCLUDE) $(DEFINES)
INCLUDE = -I.
COMPOPT =
DEFINES = -D_POSIX_SOURCE -D_XOPEN_SOURCE
CC      = /bin/xlc
LD      = /bin/ld
RM      = /bin/rm

SRC     = 850_88591.c
TARGET  = 850_88591

ENTRY_POINT       = instantiate

$(TARGET) :
```

```
	cc -e $(ENTRY_POINT) -o $(TARGET) $(SRC) -l iconv

clean :
	$(RM) -f $(TARGET)
	$(RM) -f *.o
```

入力メソッド

グローバリゼーションがベースを提供する国際環境で実行するアプリケーションの場合は、入力メソッドが必要です。入力メソッドは、アプリケーション・プログラミング・インターフェース (API) であり、特定の言語、キーボード、またはコード・セットに依存しないアプリケーションを開発できるようにします。

入力メソッドの各タイプには、次のような機能があります。

入力メソッドのタイプ	機能
キー・マップ	入力メソッドと連動してサポートされているロケールを判別する、入力メソッド・キー・マップ群 (imkeymaps)。
キー・シンボル	入力メソッドで処理できるキー記号群 (keysyms)。
修飾記号	それぞれがマスク値を持ち、入力メソッドでサポートされている修飾記号または状態群。

関連概念:

2 ページの『入力メソッドのサポート』
文字セットが大規模な言語の場合は、文字の入力は複雑になります。 例えば、中国語、韓国語、および日本語は字数が多いので、文字に対するキー・ストロークについて、 1 対 1 のキー・マッピングを備えられません。しかしながら、特殊な入力メソッドを使用すると、ユーザーは文字を音声入力したりキー入力したりでき、それらの文字を固有の言語の文字に変換できます。

157 ページの『予約済みキー・シンボル』
このセクションでは、入力メソッドによって予約されたキー・シンボルについて説明します。

194 ページの『AIXwindows チェックリスト』
残りのチェックリスト項目は、AIXwindows システムに固有のものです。

入力メソッドの紹介

入力メソッドとは、キー・ストロークをそれぞれのロケールで指定したコード・セットの文字ストリングに変換する機能群です。 入力メソッド機能には、ロケール固有の入力処理制御機構およびキーボード制御機構があります (例えば、Ctrl、Alt、Shift、Lock、および Alt-Graphic)。入力メソッドを使用することにより、さまざまなタイプの入力が可能になりますが、このセクションでは、キーボード・イベントだけを扱います。

使用しているロケールにより、どの入力メソッドをロードするか、入力メソッドを実行する方法、およびどのデバイスを使用するかが決まります。 次に、入力メソッドが状態とその結果を定義します。

入力メソッドがキー・ストロークを文字ストリングに変換するときには、変換プロセスは、使用しているキーボードとコード・セットを考慮に入れます。 標準的なキーボードを持っていないか、コード・セットをカスタマイズする場合、独自の入力メソッドを作成することができます。

多くの言語では、語を形成するのに少しの記号または文字を使用します。キーボードでテキストを入力する場合、アルファベットの記号に対応するキーを押します。 アルファベットの文字がキーボードにない場合、キーを組み合わせて押す必要があります。 入力メソッドには、そのような文字を組み合わせられるようにするアルゴリズムが用意されています。

いくつかの言語では、表意文字システムが使用されています。 この場合、語を表現するのに、文字のグループではなく、固有の記号が使用されます。 例えば、中国、日本、韓国、および台湾で使用されている文字セットには、 5,000 を超える文字があります。 そして、1 つの文字を表すために複数のバイトを使用しなければなりません。 さらに、1 つのキーボードの中に、必要な表意文字記号すべてを収めることはできません。 それで、マルチバイト文字を作成する入力メソッドが必要になります。

/usr/lib/nls/loc ディレクトリーには、システムにインストールされた入力メソッドがあります。 このディレクトリーの内容をリストして、使用可能な入力メソッドを判別できます。 入力メソッドのファイル名の形式は、言語区域.**im** です。 例えば、**fr_BE.im** ファイルは、ベルギーで使用されるフランス語用の入力メソッド・ファイルです。

よく構造化されたプロトコルを利用して入力メソッドを使用すると、アプリケーションは、ロケール固有の入力処理を使用しなくても、異なる入力をサポートできるようになります。

AIX では、aixterm に入力メソッドが用意されています。 AIXwindows インターフェースから入力された文字がサーバーに到達すると、それらの文字はキー・コードの形式になります。 クライアントで提供されているテーブルは、キー・コードを、事前定義されたコード群である *keysyms* に変換します。 キーボードによって生成されたキー・コードにはすべて、keysym があります。 これらの keysym は、MIT X Consortium によって保守され割り当てられています。 keysym は、クライアントの aixterm 端末エミュレーターに渡されます。 aixterm では、入力 keysym は、入力メソッドによってファイル・コードに変換され、その後でアプリケーションに送信されます。 X サーバーは、システム・ハードウェアに付属のディスプレイ・アダプターに合わせて機能するよう設計されています。 X サーバーは、ソケット経由で X クライアントと通信します。 したがって、相互にやり取りできるのであれば、サーバーおよびクライアントは、 1 つのネットワーク内の別個のシステムに存在していても構いません。キーボードからのデータが X サーバーに入り、そのサーバーから、端末エミュレーターに渡されます。 端末エミュレーターは、そのデータをアプリケーションに渡します。 データがアプリケーションからディスプレイ・デバイスにいく場合、ソケット経由で端末エミュレーターを通過してサーバーに渡され、サーバーからディスプレイ・デバイスに渡されます。

入力メソッド名

使用可能な入力メソッド群は、インストールされているロケールと、それらのロケールで提供される入力メソッドによって異なります。入力メソッドの名前は、通常はロケールに対応しています。 例えば、ギリシャ語入力メソッドは el_GR と指定されますが、これは、ギリシャで話されているギリシャ語言語のロケールと同じです。

1 つのロケールに複数の入力メソッドが存在する場合、ロケール名の一部である修飾子により、2 次入力メソッドを識別します。 例えば、カナダで話されるようなフランス語ロケールでは、デフォルトのメソッドと 2 つの代替メソッドの 3 つの入力メソッドがあります。 入力メソッド名は、次のとおりです。

入力メソッド名	説明
fr_CA	デフォルト入力メソッド
fr_CA@im=alt	代替入力メソッド
fr_CA.im__64	64 ビット入力メソッド

ロケールの **fr** 部分は、言語名 (フランス語) を表し、 **CA** は地域名 (カナダ) を表します。 **@im=alt** 文字ストリングは、ロケールの修飾子部分で、代替入力メソッドを識別するのに使用されます。 修飾子文字ストリングはすべて、**@im=**修飾子 の形式で示されます。

入力メソッドはロード可能なオブジェクト・モジュールであるため、 64 ビット環境で実行するときには、別のオブジェクトが必要になります。 64 ビット環境では、入力メソッド・ライブラリーは、入力メソッドを検索するときに、自動的に名前へ **__64** を追加します。 前述の例では、入力メソッドの名前は **fr_CA.im__64** になります。

ロケール名を使用しないで入力メソッドを指定することが可能です。 **libIM** ライブラリーでは、名前がロケール名に限定されていないので、呼び出し側アプリケーションでは、 **libIM** に渡された名前を見つけられなければなりません。 しかし、アプリケーションは、**@im=**修飾子 の形式の修飾子文字ストリングだけを要求する必要があり、ユーザーの要求は **setlocale** サブルーチンからの戻り文字ストリング (**LC_CTYPE,NULL**) と連結されるはずです。

入力メソッド領域

複雑な入力メソッドでは、ユーザーとの直接会話が必要になります。 例えば、日本語入力メソッドでは、入力するキーの音声上の一致に基づき、候補文字ストリングのメニューを表示しなければならない場合があります。

キー・ストロークのフィードバックは、画面上の 1 つ以上の領域に示されます。 入力メソッド領域は、次のとおりです。

Status (状況**)**
「Status (状況)」領域には、テキスト・データとビットマップを表示できます。 「Status (状況)」領域は、キーボード上の発光ダイオード (LED) の拡張機能です。

Pre-edit (事前編集**)**
クライアントがデータを処理する前に、構成される言語のために、「Pre-edit (事前編集)」領域に中間テキストが表示されます。

入力メソッドの共通の機能は、キーの組み合わせを押して、 1 つの文字または文字群を表すことです。 キー・ストロークによって文字を構成するこのプロセスのことを、事前編集 と言います。

補助
入力メソッドをカスタマイズできるようにするメニューとダイアログが、「Auxiliary (補助)」領域に表示されます。 複数の「Auxiliary (補助)」領域を、入力メソッドに管理させ、クライアントから独立させることができます。

入力メソッド領域の管理は、アプリケーション (またはツールキット) と入力メソッドとの間で役割を分割することを基本としています。 役割は、次のように分けられます。

- アプリケーションは、入力メソッド領域のサイズと位置を担当します。
- 入力メソッドは、入力域の内容を担当します。入力メソッド領域側で配置を提案することはできません。

入力メソッドのコマンド

入力メソッドは、一群のサブルーチンで、キー・ストロークを、ロケールで指定したコード・セットの文字ストリングに変換します。 入力メソッド・サブルーチンには、ロケール固有の入力処理およびキーボード制御 (Ctrl、Alt、Shift、Lock、Alt Graphic) のロジックが含まれています。

次のコマンドを使用すると、入力メソッド・サブルーチンを使用するための入力メソッド・マッピングをカスタマイズできます。

keycomp
キーボード・マッピング・ファイルをコンパイルして入力メソッド・キー・マップ・ファイルにします。

プログラミング入力メソッド

入力メソッドは、各国語サポートで提供される国際環境でアプリケーションを実行できるようにするプログラミング・インターフェースです。

入力メソッドには、次のような特性があります。

- ローカライズされた入力のサポート (ロケールによって定義される)
- 複数のキーボードのサポート
- マルチバイト文字入力処理

注: 物理キーボードが使用中であることを前提にしないでください。 ロケール設定を基にして入力メソッドを使用し、キーボード入力を処理してください。

初期化

このセクションでは、入力メソッドの判別に使用されるサブルーチンについて説明します。

IMQueryLanguage サブルーチンを使用して、入力メソッドが初期化しないでも使用できるかどうかを判別できます。 アプリケーション (ツールキット) は、ロケール固有の入力メソッド・エディター (IMED) を初期化する、 IMInitialize サブルーチンを呼び出すことにより、ロケール固有の入力メソッドを初期化します。 このサブルーチンは、**LOCPATH** 環境変数を使用して、 **LANG** 環境変数で指定した入力メソッドを検索します。 **LOCPATH** 環境変数は、入力メソッドの検索に使用するディレクトリー名群を指定します。

入力メソッドが検出されると、**IMInitialize** サブルーチンは **load** サブルーチンを使用して、入力メソッドをロードし、**imkeymap** ファイルを添付します。 入力メソッドにアクセスすると、タイプ **IMFep** (入力メソッド・フロントエンド・プロセッサー) のオブジェクトが戻されます。 **IMFep** は、隠された構造として扱われる必要があります。

各 **IMFep** は、IMInitialize サブルーチンが呼び出される場合に、ロケールのコード・セットを継承します。 したがって、**IMFilter** および **IMLookupString** サブルーチンによって戻される文字ストリングは、ロケールのコード・セットに含まれます。 **IMInitialize** サブルーチンが呼び出された後にロケールを変更することは、 **IMFep** のコード・セットに影響しません。

アプリケーションは、**IMFep** ごとに、 **IMCreate** サブルーチンを使用して、 1 つ以上の **IMObject** インスタンスを作成できます。 **IMObject** は、それ自体の状態を管理し、複数の入力メソッド領域を管理できます (128 ページの『入力メソッド領域』を参照)。 各 **IMObject** が入力処理を定義する方法は、コード・セットと、ロケールに関連付けられているキーボードに依存しています。 一番簡単なケースとしては、アプリケーションがユーザーとの 1 つの会話を管理している場合に、 1 つの **IMObject** が必要にな

ることを挙げられます。 入力メソッドは、他のユーザー・インターフェースもサポートしていますが、このようなインターフェースでは、アプリケーションがユーザーとの複数の会話を許可し、各会話には 1 つの **IMObject** が必要になります。

IMFep と **IMObject** の相違点は、 **IMFep** はアプリケーションを入力メソッドのコードにバインドするハンドルであるのに対して、 **IMObject** はキーボードなどの入力デバイスの状態のインスタンスを表すハンドルである点です。 **IMFep** は、入力メソッドの状態は表しません。 各 **IMObject** は、特定の入力状態に初期設定され、受信するイベントのシーケンスに応じて変更されます。

IMObject を作成したら、アプリケーションはキー・イベントを処理できます。 アプリケーションは、**IMFilter** および **IMLookupString** サブルーチンを使用して、キー・イベントを **IMObject** に渡す必要があります。 これらのサブルーチンは、IMED の内部処理をカスタマイズされたキー・イベント・マッピング・プロセスから分離するために用意されています。

入力メソッドの管理

このセクションでは、保守のために使用されるサブルーチンについて説明します。

入力メソッドは、保守のために以下のサブルーチンを用意しています。

入力メソッド	説明
IMInitialize	指定した言語の標準入力メソッドを初期化します。 ハンドルを、ロケールに関連付けられた IMED に戻します。ハンドルは、タイプ IMFep の隠された構造体です。
.**IMQueryLanguage**	指定された言語のサポートの有無を検査します。
IMCreate	特定の入力メソッドの 1 つのインスタンスを作成する。このサブルーチンは、キー・イベント処理を実行する前に呼び出す必要があります。
IMClose	入力メソッドをクローズします。
IMDestroy	入力メソッドのインスタンスを破壊します。

入力メソッドのキー・マップ管理

入力メソッドには、キー・イベントを文字ストリングにマップする、複数のサブルーチンが備えられています。 このマッピングは、**LOCPATH** ディレクトリーにある **imkeymap** ファイルで保守されます。

マッピングに使用するサブルーチンは、次のとおりです。

サブルーチン	説明
IMInitializeKeymap	指定された言語に関連する imkeymap を初期化します。
IMFreeKeymap	**IMInitializeKeymap** サブルーチンによって割り当てられたリソースを解放します。
IMAIXMapping	キー記号および状態パラメーターの組を文字ストリングに変換し、その文字ストリングへのポインターを戻します。
IMSimpleMapping	キー記号および状態パラメーターの組を文字ストリングに変換し、その文字ストリングへのポインターを戻します。

キー・イベントの処理

入力処理は、キーボードのキーを押すと開始します。

アプリケーションは、これらの機能を呼び出す前に、 **IMObject** を作成しておく必要があります。

表 10. キー・イベントの処理

機能	説明
IMFilter	キー・イベントが内部的に使用されているかどうかを示すよう IMED に依頼します。IMED がローカライズされた文字ストリングを作成する場合、キー・イベントはその文字ストリングにマップされます。
IMLookupString	キー・イベントをローカライズされた文字ストリングにマップします。
IMProcessAuxiliary	入力メソッドに補助域への入力を通知します。
IMIoctl	入力メソッドで、さまざまな制御操作または照会操作を行います。

コールバック

IMED は、入力メソッド・コールバック (IM-CB) API を使用して、アプリケーションに備えられたグラフィック従属機能 (コールバック) にアクセスすることにより、ユーザーと直接にやり取りします。 アプリケーションは、コールバックを追加します。これは、初期設定時に、 **IMObject** に対して出力機能を実行し、情報を照会します。 アプリケーションは、そのまますべての入力を処理します。

IMED がユーザーとやり取りするのに使用するコールバック機能群は、呼び出し側で提供しなければなりません。 IM-CB API で定義されたサブルーチンの説明は、 133 ページの『コールバックの使用』を参照してください。

入力メソッドの構造

このセクションでは、大構造入力メソッドについて説明します。

入力メソッドによって使用される大構造は、次のとおりです。

構造	説明
IMFepRec	フロントエンド情報を含みます。
IMObjectRec	入力メソッド・オブジェクトの共通部分が入っています。
IMCallback	コールバック・サブルーチンを **IMFep**に登録します。
IMTextInfo	テキスト域についての情報、主に事前編集文字ストリングについての情報を含みます。
IMAuxInfo	補助領域の内容と要求された処理のタイプを定義します。
IMIndicatorInfo	標識の現在の値を示します。
IMSTR	ヌル終了しない文字ストリングを指定します。
IMSTRATT	ヌル終了しない文字ストリングとその属性を指定します。

キーボード・マッピングの処理

以下のモデルで、アプリケーションが入力メソッドを使用する方法を示します。 この情報を使用して、キーボード・マッピングのカスタマイズに役立ててください。

入力処理は、次の 3 つのステップに分かれます。

1. **keycode/keystate(raw) - > keysym/modifier(new) (**キー・コード**/**キー状態 **(**未加工**) - >** キー・シンボル**/**修飾記号 **(**新規**))**

このステップは、アプリケーションおよび環境によって決まります。 入力メソッドへの入力のためのキー・シンボル/修飾記号に対する未加工のキー・イベントのマッピングは、アプリケーションが行います。

AIXwindows 環境では、クライアントは、サーバーでインストールされた、サーバーのキー・シンボル・テーブル **xmodmap** を使用して、このステップを行います。 **xmodmap** は、「Shift」、「Lock」、および「Alt-Graphic」キーのマッピングを定義します。 クライアントは、**xmodmap** に加えて、 X イベントからの Shift および Lock 修飾記号を用いて、このイベントが表すキー・シンボル/修飾記号を判別します。

例えば、 **XK_a** キー・シンボルを Shift 修飾記号と一緒に押すと、 **xmodmap** はそれを **XK_A** キー・シンボルにマップします。 キー・コードをキー・シンボルにマップするのに「Shift」キーを使用したので、アプリケーションは、Shift 修飾記号を元の X イベントからマスクする必要があります。 したがって、入力メソッドへの入力は、 **XK_A** キー・シンボルであり、修飾記号はありません。

別の環境では、デバイスが追加情報を提供しない場合は、入力メソッドは Shift 修飾記号と一緒に **XK_a** キー・シンボルを受け取ります。 入力メソッドは、両方の場合に同じマッピングを行い、A の文字を戻す必要があります。

2. **keysym/modifier(new) - > localized string (**キー・シンボル/修飾記号 **(**新規**) - >** ローカライズされたストリング**)**

   このステップは、ローカライズされた IMED によって決まり、各ロケールで変わります。 このステップは、キー・イベントの発生を IMED に通知し、その IMED が内部的にキー・イベントを使用するという指示を求めます。 これは、アプリケーションが **IMFilter** サブルーチンを呼び出したときに発生します。

   キー・イベントが内部の処理に使用されることを IMED が示せば、アプリケーションはこのイベントを無視します。 このイベントを見るのは IMED が最初なので、このステップは、アプリケーションがイベントを解釈する前に行われなければなりません。 IMED は、重要なキー・イベントだけを使用します。

   イベントが内部の処理に使用されないことを IMED が示すと、アプリケーションは次のステップを実行します。

3. **keysym/modifier(new) - > customized string (**キー・シンボル/修飾記号 **(**新規**) - >** カスタマイズされたストリング**)**

   このステップは、アプリケーションが **IMLookupString** サブルーチンを呼び出したときに発生します。入力メソッド・キー・マップ (**keycomp** コマンドにより作成される) は、このフェーズのマッピングを定義します。 これは、キー・イベントをストリングにマップする最後の試みであり、ユーザーはこのマッピングをカスタマイズすることができます。

   キー・シンボル/修飾記号 (新規) の組み合わせが、入力メソッド・キー・マップ (imkeymap) で定義される場合には、ストリングが戻されます。 それ以外の場合には、キー・イベントは入力メソッドには認識されません。

入力メソッドのキー・マップ

入力メソッドはユーザー定義の imkeymap をサポートし、したがって、ユーザーは入力メソッドのマッピングをカスタマイズすることができます。 入力メソッドは、ロケールごとに imkeymap をサポートします。 imkeymap のファイル名は、 imkeymap ファイルの接尾部が **.im** ではなく **.imkeymap** である点を除き、入力メソッドのファイル名に似ています。

この例では、イタリア語の入力メソッドを使用して、 **imkeymap** ファイルのカスタマイズ方法を図示しています。

1. 次のように入力して、デフォルトの **imkeymap** 送信元ファイルを **$HOME** ディレクトリーにコピーします。

   ```
   cd $HOME
   cp /usr/lib/nls/loc/it_IT.ISO8859-1.imkeymap.src .
   ```

2. 次のように入力して、 **imkeymap** 送信元ファイルをデフォルトのファイル形式に従って編集します。

   ```
   vi it_IT.ISO8859-1.imkeymap.src
   ```

3. 次のように入力して、 **imkeymap** 送信元ファイルをコンパイルします。

   ```
   keycomp < it_IT.ISO8859-1.imkeymap.src > it_IT.ISO8859-1.imkeymap
   ```

4. 次のように入力して、 **LOCPATH** 環境変数が **$HOME** を **/usr/lib/nls/loc** の前に指定していることを確認します。

   ```
   LOCPATH=$HOME:$LOCPATH
   ```

   注**:** すべての **setuid** プログラムと **setgid** プログラムは、 **LOCPATH** 環境変数を無視します。

インバウンドとアウトバウンドのマッピング

imkeymap は、キー・シンボルをファイル・コード・セット・ストリングにマップします。 **/usr/lib/nls/loc** ライブラリーにあるローカライズされた imkeymap は、インバウンド・キーのすべてのマッピングを組み込むように定義されます。

imkeymap には、以下のタイプのマッピングがあります。

マッピング・タイプ	説明
インバウンド・マッピング	ロケールのコード・セットでエンコードされたターゲット・ストリングを生成する、キー・シンボル/修飾記号のマッピング。
アウトバウンド・マッピング	ロケールのコード・セットに組み込まれたターゲット・ストリングを生成しない、キー・シンボル/修飾記号のマッピング。

特殊な imkeymap **usr/lib/nls/loc/C@outbound.imkeymap** は、この製造業者が作成したすべてのキーボードのアウトバウンド・マッピングを定義し、主として aixterm による使用を意図しています。 この imkeymap には、PF キー、カーソル・キー、およびアプリケーションが一般的に使用するその他の特殊キーのマッピングが含まれます。 標準入力と標準出力を用いる国際化アプリケーションでは、異なるキーボードで変わらないアウトバウンド・マッピングへの依存関係を制限する必要があります。例えば、Alt-a は、この製造業者が製造するすべてのキーボードで同じ方法で定義されます。しかし、Alt-tilde は、使用するキーボードによって異なります。

aixterm は、そのアウトバウンド・マッピングのベースを **C@outbound** imkeymap に置いています。マッピングの必要がより多いアプリケーションの場合は、ローカライズされた imkeymap 送信元を変更して、必要な定義を組み込むようにする必要があります。

コールバックの使用

入力メソッドを使用するアプリケーションは、コールバック関数を提供して、入力メソッド・エディター (IMED) がユーザーと通信できるようにする必要があります。 使用する入力メソッドのタイプによって、コールバックの必要の有無が決まります。 例えば、単一バイトの入力メソッドは、コールバックを必要としませんが、日本語入力メソッドはコールバックを事前編集機能とともに頻繁に使用します。 事前編集を用いると、アプリケーションにコミットする前に文字を処理することができます。

入力メソッドを使用する場合に、事前編集データの挿入もしくは削除、およびテキストのスクロールを行えるのは、アプリケーションだけです。 したがって、キー・ストロークのエコーは、コールバックを使用した入力メソッド・ロジックからの要求があったとき、アプリケーションによって行われます。

キー・ストロークを入力すると、アプリケーションは **IMFilter** サブルーチンを呼び出します。入力メソッドは、戻る前に、新規キー・ストロークを挿入するためのエコー・コールバック関数を呼び出すことができます。 文字が組み立てられると、**IMFilter** サブルーチンはそれを戻し、そのキー・ストロークは削除されます。

いくつかのケースでは、入力メソッド・ロジックがクライアントを呼び出さなければならないことがあります。これらのケースは、それぞれコールバック・アクションによって定義されます。 クライアントは、アクションごとに呼び出すべきコールバックを指定します。

コールバックには、以下のタイプがあります。

- テキスト描画

  IMED は、テキスト・コールバックを用いて、現在組み立て中の事前編集テキストを描画します。 アプリケーションおよび IMED は、コールバックが必要なとき、編集が行われる単一行バッファーを共用します。 IMED は、コールバックがそのときユーザーに提示するカーソル情報の提供も行います。

  テキスト・コールバックを以下に示します。

コールバック・サブルーチン	説明
IMTextDraw	テキスト・ストリングの描画を、アプリケーション・プログラムに求める。
IMTextHide	テキスト域を隠すよう、アプリケーション・プログラムに指示する。
IMTextStart	事前編集スペースの長さを、アプリケーション・プログラムに通知する。
IMTextCursor	テキスト・カーソルの移動を、アプリケーション・プログラムに求める。

- インディケーター (状況)

  IMED は、インディケーター・コールバックを用いて内部状況を要求します。 **IMIoctl** サブルーチンは、 **IMQueryIndicatorString** コマンドを処理して、内部状況を提供するテキスト・ストリングを検索します。 インディケーター・コールバックは、単一行バッファーを共用するのではなく状況値が使用される点を除き、テキスト・コールバックに似ています。

  インディケーター・コールバックを以下に示します。

コールバック・サブルーチン	説明
IMIndicatorDraw	状況インディケーターを描画するよう、アプリケーション・プログラムに指示する。
IMIndicatorHide	状況インディケーターを隠すよう、アプリケーション・プログラムに指示する。
IMBeep	ビープ音を出すよう、アプリケーション・プログラムに指示する。

- 補助

  IMED は、補助コールバックを用いてユーザーとの複雑なダイアログを要求します。 したがって、これらのコールバックは、テキスト・コールバックもしくはインディケーター・コールバックより精巧です。

補助コールバックを以下に示します。

コールバック・サブルーチン	説明
IMAuxCreate	補助域を作成するよう、アプリケーション・プログラムに指示する。
IMAuxDraw	補助域を描画するよう、アプリケーション・プログラムに指示する。
IMAuxHide	補助域を隠すよう、アプリケーション・プログラムに指示する。
IMAuxDestroy	補助域を破棄するよう、アプリケーション・プログラムに指示する。

IMAuxInfo 構造体は、 IMED が必要とするダイアログを定義します。

補助域の内容は、 **/usr/include/im.h** ライブラリーにある **IMAuxInfo** 構造体によって定義されます。

サブルーチンの内容	説明
IMTitle	補助域の表題を定義する。 これはマルチバイト・ストリングです。 `title.len` が 0 ならば、表示する表題はありません。
IMMessage	表示するメッセージのリストを定義する。アプリケーションの見地からは、 **IMMessage** 構造体は、情報量の多い出力専用のテキストとして扱われるべきです。 しかし、入力メソッドによっては、**IMMessage** 構造体を用いて、ユーザーとのダイアログ (この場合、 **IMFilter** または **IMLookupString** サブルーチンを用いて受け取られたキー・イベントが入力メソッドへの入力として扱われる) を行うものもあります。 このような場合は、入力メソッドは **IMMessage** 構造体を選択可能なリストかプロンプト域として扱うことがあります。 いずれの場合も、アプリケーションは、メッセージの内容のみを表示します。 **IMSelection** 構造体に **IMPanel** 構造体が含まれていない上に、 **IMButton** フィールドがヌルである場合は、 **IMProcessAuxiliary** サブルーチンを呼び出す必要はありません。 `message.nline` は、 **IMMessage** 構造体に含まれるメッセージ数を示します。各メッセージは、単一行と見なされます。 ¥t のような制御文字は認識されません。 各メッセージのテキストは、**IMSTRATT** 構造体によって定義されます。これは、マルチバイト・ストリングと属性ストリングの両方から構成されます。 各属性は、テキスト・ストリング内のバイトごとに 1 対 1 でマップされます。 `message.cursor` が真ならば、 **IMMessage** 構造体は `message.cur_row`、 `message.cur_col` の位置のテキスト・カーソルを定義します。 `message.cur_col` フィールドはバイトで定義されます。`message.maxwidth` フィールドには、列単位で定義されるすべてのテキスト・メッセージの最大幅が含まれます。

サブルーチンの内容	説明
IMButton	ユーザーに提示できる、可能なボタンを示す。 IMButton フィールドは、エンド・ユーザーに提供すべきユーザー・インターフェース制御をアプリケーションに伝えます。 ボタン・メンバーは、int 型で、以下のマスクが入っています。 **IM_OK** 「OK (了解)」ボタンを表す。 **IM_CANCEL** 「CANCEL (取消)」ボタンを表す。 **IM_ENTER** 「ENTER (入力)」ボタンを表す。 **IM_RETRY** 「RETRY (再試行)」ボタンを表す。 **IM_ABORT** 「ABORT (異常終了)」ボタンを表す。 **IM_YES** 「YES (はい)」ボタンを表す。 **IM_NO** 「NO (いいえ)」ボタンを表す。 **IM_HELP** 「HELP (ヘルプ)」ボタンを表す。 **IM_PREV** 「PREV (前へ)」ボタンを表す。 **IM_NEXT** 「NEXT (次へ)」ボタンを表す。 アプリケーションでは、 **IMProcessAuxiliary** サブルーチンを用いてボタン選択を連絡する必要があります。
IMSelection	表意文字のように、エンド・ユーザーが選択できる、項目のリストを定義する。 この構造体は、入力メソッドが大量の項目を表示したいが、ユーザーに対するリストの表示方法は制御したくない場合に使用します。 **IMSelection** 構造体は、 **IMPanel** 構造体のリストとして定義されます。 必ずしもすべてのアプリケーションが、 **IMAuxInfo** 構造体内の **IMSelection** 構造体をサポートするとは限りません。 **IMSelection** 構造体をサポートするアプリケーションは、**IMObject** の作成後、ただちに **IMIoctl** サブルーチンを用いて **IM_SupportSelection** 操作を行う必要があります。 さらに、すべてのアプリケーションが **IMPanel** 構造体をサポートするとは限りません。 したがって、 `panel_row` および `panel_col` フィールドは、すべての入力メソッドによって 1 の設定値に限定されます。 各 **IMPanel** 構造体は、`IMItem` フィールド (次元が `item_row x item_col` として定義される、二次元の行/列リストとして扱われる必要のあるフィールド) のリストから成ります。 `item_col` が 1 の場合は、1 列のみが存在します。 **IMPanel** 構造体のサイズは、バイトで定義されます。 **IMPanel** 構造体内の各項目は `panel->maxwidth` 以下です。 アプリケーションでは、 **IMProcessAuxiliary** サブルーチンを用いて 1 つ以上のユーザー選択を連絡する必要があります。 **IM_SELECTED** 値は、選択された項目を示します。 **IM_CANCEL** 値は、ユーザーが補助ダイアログを終わらせたいことを示します。
hint	入力メソッドが、 **IMAuxInfo** 構造体のコンテキストに関する情報を提供するときに使用する。 `IM_AtTheEvent` の値は、 **IMAuxInfo** 構造体が、 **IMFilter** か **IMLookupString** のいずれかのサブルーチンによって、入力メソッドに渡された最後のイベントに関連していることを示します。ほかのヒントは、複数の **IMAuxInfo** 構造体が表示されているときにそのことを示すために使用されます。

サブルーチンの内容	説明
status	内部処理のために入力メソッドが使用する。このフィールドは、アプリケーションは使用できません。 各 **IMAuxInfo** 構造体は、他の構造体からは独立しています。 メンバーの表示に使用するメソッドは、入力メソッドの呼び出し元によって決められます。 **IMAuxInfo** 構造体は **IMAuxDraw** コールバックが使用します。

関連概念:

145 ページの『漢字の事前編集』

ローマ字仮名変換モードで操作する場合は、次の 2 つのステップを行って漢字を作成する必要があります。 まず、ユーザーは、そのローマ字の表音文字をタイプ入力することによって、ひらがなを入力します。このステップで、ひらがなの表音を組み立てる、 1 つから 3 つのローマ字英字キーをタイプ入力することによってひらがなを作成します。次に、変換キーを押すことによって、ひらがなを漢字に変換します。単一の表音文字句に関連付けられた漢字が多数生じる場合があります。 変換キーは、可能性の最も高い漢字候補を表示します。変換キーを繰り返し押すうちに、ほかの候補もすべて表示されます。

コールバックの初期化

IMCreate サブルーチンを呼び出すときは、すべてのコールバックが識別されなければなりません。 **IMCallback** 構造体には、各コールバック関数のアドレスが入っています。 **IMCreate** サブルーチンの呼び出し元は、このアドレスによって **IMCallback** 構造体を初期化する必要があります。

コールバック関数は、 **IMCreate** サブルーチンが制御を呼び出し元に戻す前に呼び出すことができます。通常は、**IMTextStart** コールバックが呼び出されて、事前編集バッファーのサイズを識別します。

両方向の入力メソッド

両方向の入力メソッド (BIM) は、アラビア語、ヘブライ語、およびウルドゥー語のキーボードを処理するようにカスタマイズされている点を除き、単一バイト入力メソッドに似ています。 BIM は、また、ヘブライ語およびアラビア語状態をラテン語状態にリンクします。 ユーザーは、「Alt」+ 右「Shift」キーを使用して、アラビア語/ヘブライ語/ウルドゥー語とラテン語層の間を切り替えることができます。 これらのキーの使用は BIM から派生します。

BIM の機能を以下に示します。

- アラビア語、ヘブライ語、およびラテン語状態をサポートする。
- ISO8859-6、ISO8859-8、IBM-1046、IBM-856、およびユニコード UTF-8 コード・セットをサポートする。
- 発音区別符号の組み立てを行う。

キー・マップ

このセクションでは、両方向の入力メソッド (BIM) でサポートされるキー・マップについて説明します。

以下の BIM に関するキー・マップがサポートされています。

- ar_AA.ISO8859-6.imkeymap
- ar_AA@alt.ISO8859-6.imkeymap
- Ar_AA.IBM-1046.imkeymap
- Ar_AA@alt.IBM-1046.imkeymap
- iw_IL.ISO8859-8.imkeymap

- iw_IL@alt.ISO8859-8.imkeymap
- Iw_IL.IBM-856.imkeymap
- Iw_IL@alt.IBM-856.imkeymap

キー設定

このセクションでは、両方向の入力メソッド (BIM) でサポートされるキー設定について説明します。

以下の BIM に関するキー設定がサポートされています。

キー設定	説明
scr-rev()	スクリーンの方向を反転し、キーボード層を新しい方向のデフォルト言語に設定する。
ltr-lang()	ラテン語キーボード層を使用可能にする。
rtl-lang()	アラビア語/ヘブライ語キーボード層を使用可能にする。
col-mod()	列見出しの調整を可能にする。この調整では、各ワードを個別の列として扱います。
auto-push()	オートプッシュ・モードを切り替える。このモードでは、左から右と右から左の混合テキストを処理します。 オートプッシュ・モードを使用できるようにすると、入力した文字もしくは選択した言語層に応じて、反転したセグメントが自動的に始まったり、終了したりします。 したがって、プッシュ関数を手動で呼び出す必要はありません。
chg-push()	プッシュ・モードを切り替える。 このモードでは、カーソルはその位置にとどまり、入力した文字をフィールドの方向とは反対の方向にプッシュします。
shp-in()	アラビア語文字をその初期の形式で形成する。
shp-is()	アラビア語文字をその分離した形式で形成する。
shp-p()	アラビア語文字をそのパススルー形式で形成する。
shp-asd()	アラビア語文字をその自動形式で形成する。
shp-m()	アラビア語文字をその中間形式で形成する。
shp-f()	アラビア語文字をその最終形式で形成する。

修飾記号

このセクションでは、両方向の入力メソッド (BIM) でサポートされる修飾記号について説明します。

以下の BIM に関する修飾記号がサポートされています。

修飾記号	説明
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

キリル文字入力メソッド (CIM)

キリル文字入力メソッド (CIM) は、キリル文字キーボードを処理するようにカスタマイズされている点を除き、単一バイト入力メソッドに似ています。

CIM の機能を以下に示します。

- キリル文字状態およびラテン語状態をサポートする。「Alt」キーと左または右の「Shift」キーを同時に押すことによって、 2 つの状態を切り替えることができます。

  注: 「Alt-Graphic (右 Alt)」キーを使用すると、各キーボード層内で、追加の文字を生成することができます。
- ロシア語ロケールおよびブルガリア語ロケールには、 101 キーおよび 102 キー・キーボード・ドライバーがサポートされる。

- ISO8859-5 コード・セットをサポートする。

キー・マップ

このセクションでは、キリル文字入力メソッド (CIM) でサポートされるキー・マップについて説明します。

以下の CIM に関するキー・マップがサポートされています。

- bg_BG.ISO8859-5.imkeymap
- mk_MK.ISO8859-5.imkeymap
- sr_SP.ISO8859-5.imkeymap
- ru_RU.ISO8859-5.imkeymap
- be-BY.ISO8859-5.imkeymap
- uk-UA.ISO8859-5.imkeymap

キー・シンボル

CIM は、**XK_CYRILLIC**、**XK_LATIN1**、および **XK_MISCELLANY** グループのキー・シンボルを使用します。

以下の予約済みキー・シンボルは、このシステムの入力メソッドに固有です。

キー・シンボル	**16** 進表記
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdiaeresis	0x180007ae

修飾記号

このセクションでは、キリル文字入力メソッド (CIM) でサポートされる修飾記号について説明します。

以下の CIM に関する修飾記号がサポートされています。

修飾記号	**16** 進表記
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

以下の CIM に関する内部修飾記号がサポートされています。

修飾記号	**16** 進表記
キリル文字層	0x20

ギリシャ語入力メソッド (GIM)

ギリシャ語入力メソッド (GIM) は、単一バイト入力メソッド (SIM) に似ていますが、ラテン語文字セットとギリシャ語文字セットの両方を処理します。これは、2 つの文字セットに対応する、キーボード・マッピングの 2 つの層もしくは状態を備えることによって行われます。

キーボードは、最初ラテン語入力状態にあります。 しかし、左側の「Alt」キーが押されたままで左の「Shift」キーが押されると、キーボードはギリシャ語入力状態に入ります。 キーボードは、左「Alt」キーが押されたままの状態で右「Shift」キーを押して、ラテン語状態に戻すことができます。 これらは、押されると状態がロックされるので、ロック「Shift」キーです。

ギリシャ語状態の間に、入力メソッドは、以下の表のように発音区別符号文字、および発音区別符号組み立てのための有効な後続文字を認識します。

表 *11.* ギリシャ語組み立て文字

キー・シンボル	有効な組み立て文字
dead_acute	大文字および小文字: アルファ、イプシロン、イータ、イオタ、オミクロン、ユプシロン、オメガ
dead_diaeresis	大文字および小文字: イオタ、ユプシロン
dead_accentdiaeresis	小文字のみ: イオタ、ユプシロン

ラテン語状態では、組み立て発音区別符号はなく、表中のキーは簡単なグラフィック文字として扱われます。

ギリシャ語および単一バイト入力メソッドは、発音区別符号組み立て順序が正しくない場合の処理方法が異なります。 このような場合は、GIM はビープ音を出して、文字を戻しません。 SIM はビープ音を出さずに、無効なキーに関連する発音区別符号文字とグラフィック文字の両方を戻します。

注**:** 「Alt-Graphic (右 Alt)」キーを使用すると、各キーボード状態内で、追加の文字を生成することができます。

キー・マップ

このセクションでは、ギリシャ語入力メソッド (GIM) でサポートされるキー・マップについて説明します。

以下の GIM に関するキー・マップがサポートされています。

- el_GR.ISO8859-7.imkeymap

キー・シンボル

GIM は、**XK_LATIN1**、**XK_GREEK**、および **XK_MISCELLANY** グループのキー・シンボルを使用します。

以下の予約済みキー・シンボルは、このシステムの入力メソッドに固有です。

キー・シンボル	**16** 進表記
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdiaeresis	0x180007ae

修飾記号

このセクションでは、ギリシャ語入力メソッド (GIM) でサポートされる修飾記号について説明します。

以下の GIM に関する修飾記号がサポートされています。

修飾記号	**16** 進表記
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

以下の GIM に関する内部修飾記号がサポートされています。

修飾記号	**16** 進表記
ギリシャ語層	0x20

日本語入力メソッド (JIM)

このセクションでは、日本語入力メソッド (JIM) について説明します。

日本語入力メソッド (JIM) には、以下の機能があります。

- ローマ字から仮名文字への変換 (RKC) をサポートする。
- 仮名文字から漢字への変換 (KKC) をサポートする。
- 半角 (半分幅) と全角 (全幅) 文字入力を組み込む。
- システム辞書およびユーザー辞書の検索を行う。
- 実行時に単語をユーザー辞書に登録する。
- コールバック関数のサポートが必須な機能 :
  - 状況および事前編集描画
  - 全候補メニュー
  - JIS 区点番号入力と IBM 漢字番号入力
- IBM-943、IBM-eucJP、および UTF-8 の各コード・セットをサポートする。内部処理の場合、JIM は IBM-943 コード・セットを使用します。ただし、JIM は、IBM-943 から変換できるコード・セット (IBM-eucJP など) はすべてサポートします。
- `/usr/lib/nls/loc/JP.im` ファイル内にある。 その他のローカライズされた入力メソッドは、すべてこのファイルに対する別名です。

カタカナおよびひらがなは、それぞれ 50 文字から構成され、仮名という表音文字のセットを形成します。日本語では、すべての音を仮名で表すことができます。

漢字は表意文字のセットです。 簡単な概念なら、単一の漢字で表すことができますが、複雑な意味は、漢字のストリングで表すことができます。 数千の漢字があります。

日本語もラテン・アルファベットを使用します。 ラテン・アルファベットは、ローマ字と呼ばれ、26 文字から構成されます。 ローマ字は普通、技術・専門分野で使用されて、日本語にはない技術用語を表します。 一般に使用される文は、カタカナ、ひらがな、漢字、ローマ字、数字、およびその他の文字が入り交じったものです。

日本語文字の処理

日本工業規格 (JIS) は、コンピューター・システムにより処理される、約 7000 の漢字を指定しています。この製造業者が製造する日本語製品は、標準文字のすべてとそれ以上をサポートしています。

文字の入力は、次の方法で行われます。

- 仮名漢字変換 (KKC)
- ローマ字仮名変換 (RKC)

表 12. 特殊な日本語キー

キー機能	キー名	機能の説明
「KKC 非変換」キー	無変換	仮名文字を現状のままにする。
「KKC 変換」キー	変換	仮名を漢字に変換する。
「KKC 全候補」キー	全候補	あり得るすべての漢字表現を表示する。
「RKC ローマ字モード」キー	ローマ字	RKC のオン/オフを切り替える。
「ひらがな Shift」キー	ひらがな	ひらがな Shift 状態になる。
「カタカナ Shift」キー	カタカナ	カタカナ Shift 状態になる。
「ローマ字 Shift」キー	英数	ローマ字 Shift 状態になる。

注**:** Shift 状態は、別の「Shift」キーを押すまで維持されます。初期状態はローマ字です。

仮名漢字変換 (KKC) テクノロジー

日本語入力メソッド (JIM) KKC テクノロジーは、あらゆる漢字もしくは漢字のセットには、表音または、カタカナもしくはひらがなで表現できる音があるという事実に基づいています。

ひらがなやカタカナは、漢字より入力が容易です。 JIM は、ひらがなおよびカタカナの表音値を分析して、それと同意義の最適の漢字を判別します。この種の表音分析は、JIM に提供される辞書とテーブルによって異なります。

入力モード

このセクションでは、日本語入力メソッドのモードについて説明します。

日本語入力メソッド (JIM) には、入力処理の制御に使用できる、以下のモードがあります。

- キーボード・マッピング

  英数字、カタカナ、またはひらがなモードの呼び出しを可能にする。
- 文字サイズ

  全角 (全幅) または半角 (半分幅) モードで入力する。
- RKC のオフ/オン

  直接仮名を入力するか、事前編集組み立てモードを呼び出して、英字と組み合わせて仮名を入力する。事前編集機能を用いると、アプリケーションにコミットする前に文字を処理することができます。

キーボード・マッピング・モードが英数字で、文字サイズ・モードが半角であれば、JIM はキーをローマ字にマップします。 このモードの組み合わせは、「英字」モードとして知られています。 事前編集は、英字モードでは必要なく、RKC モードの設定に関係なく、呼び出すことはできません。 その他のモードの組み合わせでは、事前編集が始まり、これらのモードで生成される文字は ASCII ではありません。

JIM による仮名漢字変換を行う場合は、以下のキーが使用されます。

キー・シンボル	キーボード・マッピング
Katakana (カタカナ)	カタカナ・シフト
Eisu_toggle (英数)	英数字シフト
Hiragana (ひらがな)	ひらがなシフト

キー・シンボル	文字サイズ
Zenkaku_Hankaku (全角/半角)	全角と半角の切り替え
Hankaku (半角)	半角 (半幅)
Zenkaku (全角)	全角 (全幅)

キー・シンボル	**RKC** のオン/オフ
Alt-Hiragana (Alt- ひらがな)	ローマ字仮名変換を使用可能/使用不可にする
Romaji (ローマ字)	* 同じ効果

* 製造業者によって異なるキー・シンボル

以下のキーも、JIM が漢字ストリングを事前編集するときに使用されます。

キー・シンボル	漢字の事前編集
Muhenkan (無変換)	変換なし - 仮名をコミット
Henkan (変換)	変換 - 次の候補を入手
Kanji (漢字)	変換と同じ
BunsetsuYomi (文節読み)	* 句を逆に移動
MaeKouho (前候補)	前の候補に移動
LeftDouble (左 2 文字)	* カーソルを 2 文字左に移動
RightDouble (右 2 文字)	* カーソルを 2 文字右に移動
ErInput (事前入力)	* 現行の事前編集ストリングを破棄

キー・シンボル	補助事前編集
Alt-Henkan (Alt- 変換)	全候補
Touroku (登録)	実行時登録
ZenKouho (全候補)	* 全候補 (同じ効果)
KanjiBangou (漢字番号)	* 漢字番号入力
HenkanMenu (変換メニュー)	* 変換モードの変更

* 製造業者によって異なるキー・シンボル

キーボード・マッピング

キーボード・マッピングには、英数字 (ローマ字)、カタカナおよびひらがなの状態があります。 各状態は、「ロック Shift」キーとして働くキー・シンボルによって呼び出されます。 キー・シンボルは、Katakana (カタカナ)、Eisu_toggle (英数)、および Hiragana (ひらがな) のシフトです。

これらのキー・シンボルのいずれかが押されると、キーボード・マッピングは、そのキーに関連する状態に入ります。 この状態は、ほかのキー・シンボルのいずれかが押されるまで維持されます。 最初のシフト状態は Eisu_toggle ですが、これはカスタマイズによって変更することができます。

ひらがなまたはカタカナ状態を呼び出すと、各キーは、それぞれの文字セット内の表音文字にマップされます。 例えば、*q* を押すと、ひらがなシフト状態のときは「ta (た)」と発音するひらがなが作られ、カタカナ・シフト状態のときは「ta (た)」と発音されるカタカナが作られ、あるいは Eisu_toggle シフト状態のときは、ローマ字の *q* が作られます。 日本語 IBM キーボードでは、キー・トップにこの 3 つのシンボル全部が表示されています。

また、キーボード・マッピングがひらがな状態にあると、入力メソッドは、各 Hiragana (ひらがな) 文字を漢字に変換できる組み立て事前編集モードに自動的に入ります。

キーによっては、ひらがなまたはカタカナを 2 つ割り当てるものもあります。 例えば、7 のキーは、ともに「ya (や)」と発音される、大小のひらがなを持ちます。 漢字、ひらがな、およびカタカナには上段および下段がないので、これらの文字は、お互いに他方と同等の上段文字あるいは下段文字というものではありません。 小さい文字は特殊な発音を表現するのに使用されます。 これらの文字は、「Shift」キーを用いて区別することができます。

文字サイズ

日本語文字セットのサブセットは、全角および半角の両方で表されます。 漢字表意文字は、全角が普通です。 表音文字および ASCII 文字は、全角と半角の両方で表現します。 ユーザーは、「Zenkaku_Henkaku」キー・シンボルを押して全角と半角を切り替えることによって、文字サイズを制御します。

ローマ字仮名変換 (RKC)

英数字キーボードに慣れたユーザーは、ひらがなやカタカナの文字より発音でタイプ入力するほうが簡単です。 JIM にはローマ字仮名変換 (RKC) があって、ユーザーは、英数字キーボード上で、ひらがなもしくはカタカナの表音で入力することができます。

漢字の事前編集

ローマ字仮名変換モードで操作する場合は、次の 2 つのステップを行って漢字を作成する必要があります。 まず、ユーザーは、そのローマ字の表音文字をタイプ入力することによって、ひらがなを入力します。このステップで、ひらがなの表音を組み立てる、 1 つから 3 つのローマ字英字キーをタイプ入力することによってひらがなを作成します。次に、変換キーを押すことによって、ひらがなを漢字に変換します。単一の表音文字句に関連付けられた漢字が多数生じる場合があります。 変換キーは、可能性の最も高い漢字候補を表示します。変換キーを繰り返し押すうちに、ほかの候補もすべて表示されます。

例えば、表音「k-a-n-j-i」に対する漢字を入力するには、次の 2 つのことを行う必要があります。

1. キーボード・マッピングをひらがな状態に設定する。
2. 「Alt- ひらがな」キーを押して、ローマ字仮名のマッピングを可能にする。このアクションで、英数字キーボードを呼び出します。

ここで、「kanji」というスペルでキーを押すことができます。 各表音が全部揃うと、ひらがなが表示されます。

ひらがなは、目で見えるフィードバックで表示され、 JIM が事前編集状態で組み立てていることを示します。 文字には下線が付き、反転表示されます。 このフィードバック機能は、コールバック として知られています。

事前編集ストリング内のひらがなを漢字に変換するには、変換キーを押します。 表音ひらがなに関連する最も可能性の高い候補が表示されます。 このキーを繰り返し押して、ほかの候補を表示します。

組み立て処理の際、事前編集ストリングは、漢字ワードと考えられるセグメントに分割されます。 仮名文字のストリングは、候補に変換されると、これらの変換可能なセグメントの 1 つとして扱われます。 事前編集ストリングが表示されている間に、 JIM はカーソル・キーと他のキーを用いてそのストリングを操作します。

事前編集ストリングをプログラムにコミットするには、ユーザーは「Enter」キーを押します。 この場合、「Enter」キー・コード自体はプログラムに送られず、ストリングのみが送られます。

事前編集をオフにし、ひらがなまたはカタカナを直接プログラムにコミットする場合は、「Muhenkan」キー・シンボルを使用することもできます。

以下の表は、シフト状態遷移、ならびに RKC モード・キーとシフト状態の相互作用を示します。

文字エンコード	コード・ポイント	説明	カウント
000xxxxx	00–1F	制御	32
00100000	20	スペース	1
0xxxxxxx	21–7E	7 ビット ASCII	94
01111111	7F	削除	1
10000000	80	未定義	1
100xxxxx 01xxxxxx	[81–9F] [40–7E]	2 バイト	1953
100xxxxx 1xxxxxxx	[81–9F] [80–FC]	2 バイト	3844
10100000	A0	未定義	1
1xxxxxxx	A1–DF	8 ビット単一バイト	63
111xxxxx 01xxxxxx	[E0–FC] [40–7E]	2 バイト	1827
111xxxxx 1xxxxxxx	[E0–FC] [80–FC]	2 バイト	3596
11111101	FD	未定義	1
11111110	FE	未定義	1
11111111	FF	すべて 1	1

JIM 内には、次のタイプの補助域があります。

- 「All Candidates (全候補)」メニュー
- 「Kanji Number Input (漢字番号入力)」ダイアログ
- 「Conversion Mode (変換モード)」メニュー
- 「Runtime Registration (実行時登録)」ダイアログ

ひらがなまたはカタカナのストリングで仮名漢字変換操作を行うと、 1 から 100 の漢字候補が出てきます。最悪の場合、正しい漢字を入手するまでに、変換キーを 100 回より多く押さなければならない場合もあり得ます。

このような場合に、正しい文字を見つけるのに便利なのは、「ZenKouho」または「Alt-Henkan」キー・シンボルで、「All Candidates (全候補)」メニューを要求することです。 このメニューは、現行ターゲット (カーソルが事前編集域で指し示している漢字ワード) に、関連する代替候補が複数あると表示されます。メニューには、選択に対する複数の候補が入っています。 「Reset」キー・シンボルを押すか、「Enter」キーを押すか、あるいは候補を選択すると、「All Candidates (全候補)」メニューが消えます。

「Kanji Number Input (漢字番号入力)」ダイアログから、 3 桁から 5 桁の数字を入力して漢字を選択するよう、ユーザーにプロンプトが出されます。 この数字は、文字のコードを表します。 オンライン辞書によってユーザーはそのコードを検索することができます。 これらの辞書の配列の形式はいろいろあります。 例えば、辞書から表音別のコードがリストされるものもあります。 また、文字の組み立てに使用する

画数 (ストローク数) で、コードを配列している辞書もあります。 「KanjiBangou」キー・シンボルは、このメニューを呼び出します。 メニューは、「Reset」キー・シンボルか「Return」キー・シンボルによって終了します。

「HenkanMenu」キー・シンボルは、「Conversion Mode (変換モード)」メニューを呼び出します。 4 つの選択項目が表示されます。 最も重要な項目は、ワード変換モードと語句変換モードです。 番号を選んで選択を行い、「Return」キー・シンボルを押します。 このメニューは、選択が行われるか、「Reset」キー・シンボルが押されると、終了します。

ユーザー辞書にストリングのマッピングを登録するための、仮名ストリングおよび漢字ストリングを入力するよう、実行時登録ダイアログからユーザーにプロンプトが出されます。このペアが登録されると、JIM はそれを変換候補として使用することができます。 メニューは、「Escape」キー・シンボルか「Reset」キー・シンボルによって終了します。

メニューの表示は、JIM が作動するインターフェース環境によって異なります。 例えば、「Page Down」キーと「Page Up」キーを使用するスクロール・メニューをサポートするインターフェースもあります。

関連概念:

133 ページの『コールバックの使用』
入力メソッドを使用するアプリケーションは、コールバック関数を提供して、入力メソッド・エディター (IMED) がユーザーと通信できるようにする必要があります。 使用する入力メソッドのタイプによって、コールバックの必要の有無が決まります。 例えば、単一バイトの入力メソッドは、コールバックを必要としませんが、日本語入力メソッドはコールバックを事前編集機能とともに頻繁に使用します。 事前編集を用いると、アプリケーションにコミットする前に文字を処理することができます。

キー・マップ

このセクションでは、日本語入力メソッド (JIM) によってサポートされるキー・マップについて説明します。

JIM では以下のキー・マップがサポートされています。

- ja_JP.IBM-eucJP.imkeymap
- ja_JP.IBM-eucJP@alt.imkeymap
- Ja_JP.IBM-943.imkeymap
- Ja_JP.IBM-943@alt.imkeymap
- JA_JP.UTF-8.imkeymap
- JA_JP.UTF-8@alt.imkeymap

キー・シンボル

JIM は、**XK_KATAKANA**、**XK_LATIN1**、および **XK_MISCELLANY** グループのキー・シンボルを使用します。

以下の予約済みキー・シンボルは、このシステムの入力メソッドに固有です。

キー・シンボル	16 進表記	説明
XK_BunsetsuYomi	0x1800ff05	語句から読みへ戻る
XK_MaeKouho	0x1800ff04	前の候補
XK_ZenKouho	0x1800ff01	全候補
XK_KanjiBangou	0x1800ff02	漢字番号入力
XK_HenkanMenu	0x1800ff03	変換モードの変更
XK_LeftDouble	0x1800ff06	カーソルを 2 文字左に移動
XK_RightDouble	0x1800ff07	カーソルを 2 文字右に移動
XK_LeftPhrase	0x1800ff08	将来の使用に予約済み
XK_RightPhrase	0x1800ff09	将来の使用に予約済み
XK_ErInput	0x1800ff0a	現行の事前編集ストリングを破棄
XK_Resetreset	0x1800ff0b	リセット

キー・シンボル	説明
XK_Kanji	ひらがなから漢字へ変換する。
XK_Muhenkan	変換を取り消す。
XK_Romaji	JIM をローマ字入力モードにする。
XK_Hiragana	JIM をひらがな入力モードにする。
XK_Katakana	JIM をカタカナ入力モードにする。
XK_Zenkaku_Hankaku	全角文字入力モードと半角文字入力モード間を切り替える。
XK_Touroku	ワードをユーザー辞書に登録する。
XK_Eisu_toggle	JIM を英数字の入力モードにする。

修飾記号

このセクションでは、日本語入力メソッド (JIM) でサポートされる修飾記号について説明します。

JIM では以下の修飾記号がサポートされています。

修飾記号	16 進表記
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

JIM では以下の内部修飾記号がサポートされています。

修飾記号	**16** 進表記
Kana (仮名)	0x20
Romaji (ローマ字)	0x40

韓国語入力メソッド (KIM)

このセクションでは、韓国語入力メソッド (KIM) について説明します。

韓国語 EUC コード・セットは、以下の主要な文字グループから構成されます。

- ASCII (英語)
- ハングル (韓国語文字)

ハングル・コード・セットには、ハングル文字と漢字 (韓国語) 文字 (元は中国語) が組み込まれています。ハングルの 1 文字は、複数の子音と母音で構成されることがあります。 しかし、ほとんどのハングルの単語は、漢字 (韓国語) で表現できます。 各漢字 (韓国語) には、その独自の意味があり、したがって、ハングルより特定度が高くなります。

現在の韓国語の標準コード・セット KSC5601 には、8224 文字のハングル、漢字 (韓国語)、および特殊文字が収められています。 韓国語標準拡張 UNIX コード (EUC) に準拠するために、このコード・セットが EUC の CS1 に割り当てられています。

文字の入力は、以下に示すようにして行うことができます。

- ASCII

  英字の入力には、ASCII モードが使用されます。
- ハングル語

  「XK_Hangul」キーは、ハングル語モードを呼び出します。このモードは、ハングル文字を入力するとき使用してください。 ハングル語モードが呼び出されると、KIM は、入ってくる子音および母音をハングル語合成規則に従って組み立てます。 ハングル文字は、子音とそれに続く母音で構成されます。 最終子音はオプショナルです。 入力されてきた文字が構成規則に違反していると、警告のビープ音が出ます。

  標準コード・セットには、約 1500 文字の特殊文字があります。 これらの文字は、KIM のコード入力関数を用いて入力する必要があります。 「Code Input (コード入力)」キーでコード入力関数を呼び出します。 コード入力関数が呼び出されると、希望する文字のコード・ポイントを、「Code Input (コード入力)」補助ウィンドウに入力することができます。
- 漢字 (韓国語)

  「XK_Hangul_Hanja」キーで、漢字 (韓国語) モードを呼び出します。 漢字 (韓国語) を変換できるのは、該当するハングル文字からのみです。ハングル漢字 (韓国語) 変換 (HHC) には、単一候補と複数候補の 2 つのモードがあります。 この場合、候補とは、あり得る文字選択肢の中から 1 つを選択することを意味します。

  単一候補モードでは、コマンド・ライン上に候補が 1 つずつ表示されます。 複数候補モードでは、補助ウィンドウに、一度に 10 候補まで表示されます。

漢字 (韓国語) 変換モードを使用しているときは、「Conversion」キーが押されたときに、ハングル文字を漢字 (韓国語) に変換することができます。 同様に、漢字 (韓国語) ワードを該当するハングル・ワードに変換することができます。

漢字 (韓国語) の入力も、ハングルの入力のときと同じ方法で、コード入力関数によって行うことができます。

これらの変換を可能にするために、 106 キーの韓国語キーボードには以下の特殊キーがあります。

表 *13.* 特殊な韓国語キー

キー機能	キー・シンボル	機能の説明
「ハングル/英字」切り替えキー	XK_Hangul	ハングルと英字モード間を切り替える
「漢字 (韓国語)」切り替えキー	XK_Hangul_Hanja	漢字 (韓国語) モードのオン/オフを切り替える
「コード入力」キー	XK_Hangul_ Codeinput	コード入力関数を呼び出す。これで、文字をそのコード・ポイントによって入力することができる。
「HHC 全候補」キー	XK_Hangul_ MultipleCandidate	複数候補モードを呼び出す
「HHC 変換」キー	XK_Hangul_ Conversion	単一候補モードを呼び出し、かつ単一候補と複数候補の両方のモードで候補の前方スクロールも行う。
「HHC 非変換」キー	XK_Hangul_ NonConversion	候補を後方スクロールする

ラトビア語入力メソッド (LVIM)

ラトビア語入力メソッド (LVIM) は、ラトビア語キーボードを処理するようにカスタマイズされている点を除き、単一バイト入力メソッド (SIM) に似ています。

LVIM の機能を以下に示します。

- QWERTY および人間工学グループを、2 つの主要グループとしてサポートする。 両方の主要グループから、デッド・キーによってアクセス可能な、さらに 2 つの補足的なグループがあります。
  - 左「Alt」キーと左「Shift」キーを同時に押すと、キーボードは人間工学グループに入ります。
  - 左「Alt」キーと右「Shift」キーを同時に押すと、キーボードは QWERTY グループに入ります。
- IBM-921 コード・セットをサポートする。

キー・マップ

このセクションでは、ラトビア語入力メソッド (LVIM) によるキー・マップのサポートについて説明します。

LVIM では以下のキー・マップがサポートされています。

- Lv_LV.IBM-921.imkeymap

リトアニア語入力メソッド (LTIM)

リトアニア語入力メソッド (LTIM) は、リトアニア語キーボードを処理するようにカスタマイズされている点を除き、単一バイト入力メソッド (SIM) に似ています。

LTIM の機能を以下に示します。

- プログラム化グループおよびリトアニア語グループを、 2 つの主要グループとしてサポートする。両方の主要グループから、デッド・キーによってアクセス可能な、さらに 2 つの補足的なグループがあります。
  - 左「Alt」キーと左「Shift」キーを同時に押すと、キーボードはリトアニア語グループに入ります。
  - 左「Alt」キーと右「Shift」キーを同時に押すと、キーボードはプログラム化グループに入ります。
- IBM-921 コード・セットをサポートする。

キー・マップ

このセクションでは、リトアニア語入力メソッド (LTIM) によるキー・マップのサポートについて説明します。

LTIM では以下のキー・マップがサポートされています。

- Lt_LT.IBM-921.imkeymap

タイ語入力メソッド (THIM)

タイ語入力メソッドは、タイ語を処理するようにカスタマイズされている点を除き、単一バイト入力メソッド (SIM) に似ています。

詳しくいえば、タイ語では無効なタイ語文字の組み合わせ (子音、上下母音、トーン・マーク) が入力されないように設計されています。 THIM の機能を以下に示します。

- ラテン語グループとタイ語グループを、キーボード上の 2 つの主要グループとしてサポートする。
  - 左「Alt」キーと左「Shift」キーを押すと、キーボードはタイ語グループに入ります。
  - 左「Alt」キーと右「Shift」キーを押すと、キーボードはラテン語グループに入ります。
- TIS-620 コード・セットをサポートする。

キー・マップ

このセクションでは、タイ語入力メソッド (THIM) によるキー・マップのサポートについて説明します。

THIM では以下のキー・マップがサポートされています。

- th_TH.TIS-620.imkeymap

ベトナム語入力メソッド (VNIM)

ベトナム語入力メソッド (VNIM) は、ベトナム語を処理するようにカスタマイズされている点を除き、単一バイト入力メソッド (SIM) に似ています。

詳しくいえば、ベトナム語では無効なベトナム語文字の組み合わせ (トーン・マーク) が入力されないように設計されています。 ベトナム語のトーン・マーク文字を入力できるのは、ベトナム語の母音 (a、e、i、o、u、y、a-circumflex、e-circumflex、o-circumflex、a-breve、o-horn、または u-horn) のいずれかの直後に限られます。

VNIM は、事前組み立て文字やベトナム語トーン・マークなどの、単一キーボード層をサポートします。

VNIM は IBM-1129 コード・セットをサポートします。

キー・マップ

このセクションでは、ベトナム語入力メソッド (VNIM) によるキー・マップのサポートについて説明します。

VNIM では以下のキー・マップがサポートされています。

- Vi_VN.IBM-1129.imkeymap

中国語 (簡体字) 入力メソッド (ZIM-UCS)

このセクションでは、中国語 (簡体字) 入力メソッド (ZIM-UCS) について説明します。

UCS-2 コード・セットは、ほとんどすべての文字グループから構成されます。 ZH_CN ロケールには、以下の文字グループがあります。

- ASCII (英語)
- グリフ
- 中国語、日本語、および韓国語 (CJK) の文字 (単一化文字)

CJK 文字セットには、20,992 の文字位置がありますが、中国語文字には 20,902 位置しか割り当てられていません。

中国語 (簡体字) の発音は、Bopomofo と呼ばれる音声シンボルによって表されます。 25 の音声シンボルがあります。 中国語 (簡体字) 文字は、1 つから 3 つの音声シンボルによって表されます。

ZIM-UCS には、以下の特性があります。

- 以下の入力メソッドが一般的に使用される。

  インテリジェント **ABC**
  : 中国語文字の表音に基づく入力メソッド。

  ピンイン入力メソッド
  : 中国語文字の表音に基づく入力メソッド。 中国語文字は、発音に従って 1 つ以上の音素に分かれます。

  Wu Bi (5 ストライク**)** 入力メソッド
  : 中国語文字の筆記に基づく入力メソッド。 WuBi 筆記の入力メソッドに従って、中国語文字はストローク、部首、および単一文字の 3 つのレベルに分類されます。

  Zheng Ma
  : 中国語ワードの筆記に基づく入力メソッド。

  Biao Xing Ma 入力メソッド
  : 中国語文字を複数の構成要素、つまり部首 に分割する入力メソッド。 文字をコーディングする際、これらの部首は、対応する英語の文字で表されます。

  内部コード入力メソッド
  : GB18030 (中国語内部コード仕様) および UCS-2 (Unicode システム・バージョン 2) に定義されているコード・テーブルに従う入力メソッド。

- 半角および全角文字入力。 単一バイト・モードおよびマルチバイト・モードの両方で ASCII 文字をサポートします。
- すべての候補リストをサポートする補助ウィンドウ。 例えば、インテリジェント ABC は、同音のシンボル (部首) を含む候補の文字のリストを生成します。 ユーザーは、変換キーを押して、希望する文字を選択します。
- オーバー・スポット事前編集描画領域。 部首を反転表示域 (テキスト行を一時的に指定する) に入力できるようにします。 変換キーを押すと、完全な文字がエディターに送られます。

UCS-ZIM ファイルは **/usr/lib/nls/loc** ディレクトリーにあります。

UCS-ZIM キー・マップは **/usr/lib/nls/loc/ZH_CN.UTF-8.imkeymap** ディレクトリーにあります。

中国語 (CJK) 文字の処理

UCS-ZIM は、いずれかの入力メソッド・キーを押すことによって呼び出されます。 キーには、各部首シンボルか音声シンボルが割り当てられています。 ユーザーは、オーバー・スポット事前編集域に部首または音声シンボルを入力します。 内部コード入力メソッドの場合は、最後のキーが押されると文字が生成されます。 その他の入力メソッドでは、ウィンドウに表示される候補のリストを生成します。 ユーザーは、候補番号を選択して、希望する文字を選びます。 無効入力には、ビープ音とエラー・メッセージが出ます。 グリフは、ABC 入力メソッドを用いて入力することができます。

単一バイト入力メソッド (SIM)

SIM は、ほとんどのロケールをサポートする標準です。これは、単一バイト・ロケールに関するワークステーション・キーボード上に定義された単純な組み合わせをサポートする、マッピング機能です。 SIM は、**keycomp** コマンドが記述できるキーボード、コード・セット、および言語であれば、いずれもサポートします。 SIM は、imkeymap を用いてカスタマイズすることができます。 入力メソッドが戻すコード化ストリングは、imkeymap によって異なります。

ほとんどの単一バイト・ロケールが 1 つの SIM をサポートします。 SIM の機能を以下に示します。

- 101 キーおよび 102 キー・キーボード・マッピングをサポートする。
- Alt と数字パッドの組み合わせをサポートする。

  「Alt」キーを押すと、入力メソッドは、次に押される 3 つの数字キーを用いて文字を組み立てます。 3 つの数字キーは、文字の 10 進エンコードを表します。 例えば、`XK_0`、`XK_9`、`XK_7` の順序で入力すると、文字 *a* (097) にマップします。
- 数字キーパッドの Num-Lock 状態をサポートする。
- 発音区別符号の組み立てをサポートする。

  「e ウムラウト」は、発音区別符号組み立ての一例です。 e ウムラウトを組み立てる場合、ユーザーは、該当する発音区別符号キー (ウムラウト) に続けて英字のキー (e) を押します。 使用される特定の発音区別符号キー (ウムラウト) は、ロケールとキーボード定義によって異なります。 発音区別符号キーの後にスペースが付くと、それがロケールのコード・セット内にあれば、そのキーで表される発音区別符号文字が戻されます。
- コールバック関数を必要としない。
- **/usr/lib/nls/loc/sbcs.im** ファイル内にある。 他のローカリゼーション入力メソッドは、ほとんどがこのファイルの別名です。

キー・マップ

このセクションでは、単一バイト入力メソッド (SIM) によってサポートされるキー・マップについて説明します。

SIM では、以下のキー・マップが使用されます。

- cs_CZ.ISO8859-2.imkeymap
- da_DK.ISO8859-1.imkeymap
- de_CH.ISO8859-1.imkeymap
- de_DE.ISO8859-1.imkeymap
- en_GB.ISO8859-1.imkeymap

- en_GB.ISO8859-1@alt.imkeymap
- en_US.ISO8859-1.imkeymap
- es_ES.ISO8859-1.imkeymap
- Et_EE.IBM-922 - imkeymap
- pl_PL.ISO8859-2@alt.imkeymap
- sq_AL.ISO8859-1.imkeymap
- fi_FI.ISO8859-1.imkeymap
- fi_FI.ISO8859-1@alt.imkeymap
- fr_BE.ISO8859-1.imkeymap
- fr_CA.ISO8859-1.imkeymap
- fr_CH.ISO8859-1.imkeymap
- fr_FR.ISO8859-1.imkeymap
- fr_FR.ISO8859-1@alt.imkeymap
- hr_HR.ISO8859-2.imkeymap
- hu_HU.ISO8859-2.imkeymap
- is_IS.ISO8859-1.imkeymap
- it_IT.ISO8859-1.imkeymap
- it_IT.ISO8859-1@alt.imkeymap
- nl_BE.ISO8859-1.imkeymap
- nl_NL.ISO8859-1.imkeymap
- no_NO.ISO8859-1.imkeymap
- pl_PL.ISO8859-2.imkeymap
- pt_BR.ISO8859-1.imkeymap
- pt_PT.ISO8859-1.imkeymap
- ro_RO.ISO8859-2.imkeymap
- sh_SP.ISO8859-2.imkeymap
- sl_SI.ISO8859-2.imkeymap
- sk_SK.ISO8859-2.imkeymap
- sv_SE.ISO8859-1.imkeymap
- sv_SE.ISO8859-1@alt.imkeymap
- tr_TR.ISO8859-1.imkeymap

予約済みキー・シンボル

以下のキー・シンボルは、この入力メソッドに固有で、 **/usr/include/X11/aix_keysym.h** ファイルに記述されています。

キー・シンボル	**16** 進値
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdieresis	0x180007ae

修飾記号

このセクションでは、単一バイト入力メソッド (SIM) によってサポートされる修飾記号について説明します。

SIM では、以下の修飾記号が使用されます。

修飾記号	**16** 進値
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10
Mod5Mask (Num Lock)	0x80

中国語 (繁体字) 入力メソッド (TIM)

このセクションでは、中国語 (繁体字) 入力メソッド (TIM) について説明します。

中国語 (繁体字) コード・セットは、以下の 2 つの文字グループから構成されます。

- ASCII (英語)
- 中国語 (繁体字) 文字

中国語 (繁体字) 文字セットには、100,000 文字を超える文字が入っていますが、頻繁に使用されるのは約 5000 文字に過ぎません。 各文字には、部首 として知られる 1 つから 5 つの部分から構成されています。

中国語 (繁体字) の発音は、 Dsu-Yin または Bo-Po-Mo-Fo と呼ばれる音声シンボルによって表されます。 37 の音声シンボルに加えて、4 つの抑揚インディケーターがあります。 中国語文字は、1 つから 3 つの音声シンボルによって表されます。 文字には、1 つの抑揚シンボルを組み込むことができます。 抑揚シンボルを省くことは、5 番目の抑揚アクセントを意味します。

TIM 機能

このセクションでは、中国語 (繁体字) 入力メソッド (TIM) の機能について説明します。

TIM には、以下の特性があります。

- 以下の入力メソッドが使用されます。

  Tsang-Jye
  文字を生成する部首をサポートする。 データ入力担当者による使用の頻度が最も高い。

  単純化 **Tsang-Jye**
  ワイルドカード入力および部首をサポートする。 この入力メソッドでも、部分文字の入力が可能です。

  音声シンボル
  文字の入力をその発音に基づいて行う。

  内部コード
  EUC 16 進数のコード・ポイント入力により文字を生成する。

  10 進値
  10 進値により文字を生成する。 いずれの入力モードからも呼び出し可能。
- 半角および全角文字入力。 単一バイト・モードおよびマルチバイト・モードの両方で ASCII 文字をサポートします。
- システム定義およびユーザー定義の可能な文字入力。
- すべての候補リストをサポートする補助ウィンドウ。 単純化 Tsang-Jye および表音入力メソッドは、同じ入力部首または音声シンボルが入っている文字候補のリストを生成します。ユーザーは、対応する番号を押して、文字を選択します。
- オーバー・スポット事前編集描画領域。 部首を反転表示域 (テキスト行を一時的に指定する) に入力できるようにします。 変換キーを押すと、完全な文字がエディターに送られます。

TIM ファイルは **/usr/lib/nls/loc/TW.im** ディレクトリー内にあります。

TIM キー・マップは **/usr/lib/nls/loc/zh_TW.IBM-eucTW.imkeymap** ディレクトリー内にあります。

中国語 (繁体字) 文字の処理

TIM は、入力メソッド・キーのいずれかを押すことによって呼び出されます。キーには、各部首シンボルか音声シンボルが割り当てられています。 ユーザーは、オーバー・スポット事前編集域に部首または音声シンボルを入力します。 Tsang-Jye および内部コード入力の場合は、変換キーが押されると文字が生成されます。 単純化 Tsang-Jye および表音文字入力では、ウィンドウに表示される候補のリストを生成します。 ユーザーは、候補番号を選択して、希望する文字を選びます。 無効入力には、ビープ音とエラー・メッセージが出ます。

中国語 (繁体字) の 106 キー・キーボードには、以下の中国語 (繁体字) 入力メソッドの特殊キーが定義されています。

表 14. 特殊な中国語 (繁体字) キー

キー機能	キー・シンボル	機能の説明
「Tsang-Jye Shift」キー	XK_Chinese _Tsangjei	Tsang-Jye および単純化 Tsang-Jye 入力メソッドの両方を呼び出す。
「表音文字 Shift」キー	XK_Chinese _Phonetic	表音文字入力メソッドを呼び出す。
「半角/全角」切り替えキー	XK_Chinese _Full_Half	半角と全角を切り替える。
「変換」キー	XK_Convert	部首シンボルおよび音声シンボル、または EUC コード・シンボルを文字に変換する。 必要に応じて、補助ウィンドウに候補リストを表示する。
「無変換」キー	XK_Non _Convert	音声シンボルを文字と解釈する。
「英数字」キー	XK_Alph_Num	ASCII モードを呼び出す。
「ALT-Tsang-Jye Shift」キー	XK_Internal _Code	内部コード入力メソッドを呼び出す。
「ALT」+ 数字キーパッド		10 進値入力メソッドを呼び出す。

汎用入力メソッド

汎用入力メソッドを Unicode/UTF-8 ロケールで使用すると、完全なマルチリンガル入力メソッドがサポートされます。

汎用入力メソッドの機能を以下に示します。

- 入力メソッドの切り替えをサポートする。
  - 「Ctrl」キー、左「Alt」および文字「i」を同時に押すと、他の使用可能な入力メソッドをリストするメニューが表示されます。 このリストから入力メソッドを選択すると、キーボードは再マップされ、指定した入力メソッドがロードされ、ロードされた入力メソッドを用いて文字入力を行えるようになります。
- 文字入力のポイントおよびクリックをサポートする。
  - 「Ctrl」キー、左「Alt」および文字「l」を同時に押すと、 Unicode 標準に入っているさまざまなカテゴリーの文字をリストしたメニューが表示されます。 文字リストを選択すると、リストから使用できる文字のマトリックスが表示されます。 指定された文字をクリックすると、その文字は入力メソッドを経由してアプリケーションに送られます。
  - 「Ctrl」キー、左「Alt」および文字「c」を同時に押すと、アプリケーションに戻ります。あるいは、すでにアプリケーションに戻っている場合は、ポイントとクリックによる文字入力で最後に使用された文字リストに戻ります。
- UTF-8 コード・セットをサポートする。

キー・マップ

これは、汎用入力メソッドの唯一のキー・マップです。

- XX_XX.UTF-8.imkeymap

予約済みキー・シンボル

このセクションでは、入力メソッドによって予約されたキー・シンボルについて説明します。

リストされたキー・シンボルは、入力メソッドが使用するために予約済みです。

キー・シンボル	**16** 進表記
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdieresis	0x180007ae
XK_BunsetsuYomi	0x1800ff05
XK_MaeKouho	0x1800ff04
XK_ZenKouho	0x1800ff01
XK_KanjiBangou	0x1800ff02
XK_HenkanMenu	0x1800ff03
XK_LeftDouble	0x1800ff06
XK_RightDouble	0x1800ff07
XK_LeftPhrase	0x1800ff08
XK_RightPhrase	0x1800ff09
XK_ErInput	0x1800ff0a
XK_Reset	0x1800ff0b

関連概念:

126 ページの『入力メソッド』
グローバリゼーションがベースを提供する国際環境で実行するアプリケーションの場合は、入力メソッドが必要です。入力メソッドは、アプリケーション・プログラミング・インターフェース (API) であり、特定の言語、キーボード、またはコード・セットに依存しないアプリケーションを開発できるようにします。

中国語 (繁体字) の予約済みキー・シンボル

このセクションでは、中国語 (繁体字) の予約済みキー・シンボルをリストします。

キー・シンボル	**16** 進表記
XK_Full_Size	0xff42
XK_Phonetic	0xff48
XK_Alph_Num	0xaff50
XK_Non_Convert	0xaff52
XK_Convert	0xaff51
XK_Tsang_Jye	0xff47
XK_Internal_Code	0xff4a

中国語 (簡体字) の予約済みキー・シンボル (ZIM および ZIM-UCS)

このセクションでは、中国語 (簡体字) の予約済みキー・シンボル (ZIM および ZIM-UCS) をリストします。

キー・シンボル	**16** 進表記
XK_Alph_Num	0xaff47
XK_Non_Convert	0xaff59
XK_Row_Column	0xaff48
XK_PinYin	0xaff49
XK_English_Chinese	0xaff50
XK_ABC	0xaff51
XK_Fivestroke	0xaff62
XK_User-defined	0xaff56
XK_Legend	0xaff55
XK_ABC_Set_Option	0xaff60
XK_Half_full	0xaff53

メッセージ機能

メッセージを、プログラムが実行時にアクセスすることができるメッセージ・カタログの形式で提供し、プログラムから分けておく必要があります。 この調整により、メッセージをさまざまな言語に翻訳し、ユーザーのロケールに基づいてプログラムで使用することが容易になります。 このタスクに役立つコマンドとサブルーチンが、メッセージ機能に備えられています。

プログラマーはアプリケーション・メッセージを含むメッセージ送信元ファイルを作成し、これらのファイルはメッセージ・カタログに変換されます。 アプリケーションは、これらのカタログを使用し、必要に応じてメッセージの検索および表示を行います。 メッセージ・ソース・ファイルを他の言語に変換し、ファイルをメッセージ・カタログに変換するのに、プログラムの変更や再コンパイルは必要ありません。

関連概念:

1 ページの『プログラムからのメッセージの分離』
メッセージをプログラムから分けておき、プログラムが実行時にアクセスすることができるメッセージ・カタログの形式で提供する必要があります。 これにより、メッセージをさまざまな言語に翻訳し、その翻訳されたメッセージをユーザーのロケールに基づいてプログラムで使用することが容易になります。

194 ページの『AIXwindows チェックリスト』
残りのチェックリスト項目は、AIXwindows システムに固有のものです。
189 ページの『プログラム操作のチェックリスト』

メッセージ・ソース・ファイルの作成

メッセージ機能には、外部化されたメッセージ・カタログにあるプログラム・メッセージを検索し、表示するためのコマンドやサブルーチンがあります。 プログラマーは、アプリケーション・メッセージが入っているメッセージ・ソース・ファイルを作成し、それを **gencat** コマンドを用いてメッセージ・カタログに変換します。

メッセージ・テキスト・ソース・ファイルを作成するには、テキスト・エディターを用いてファイルをオープンします。 メッセージ識別番号かシンボル ID を入力します。その後、次の例に見られるようなメッセージ・テキストを入力します。

```
1 message-text   $       (This message is numbered)
2 message-text   $       (This message is numbered)
OUTMSG message-text  $   (This message has a symbolic identifier ¥
                          called OUTMSG)
4 message-text   $       (This message is numbered)
```

使用上の考慮事項

このセクションでは、使用上の考慮事項について説明します。

以下のことについて考慮します。

- メッセージの ID 番号 (または ID) と、メッセージ・テキストの間に、ブランク文字を 1 つ置く必要がある。
- シンボル ID は、英字から始まり、かつ英字、10 進数、および下線のみ入れることができる。
- シンボル ID の先頭文字を、数字にすることはできない。
- シンボル ID の最大長は 64 バイトである。
- メッセージ ID 番号は、単一のメッセージ・セット内で昇順に割り当てる必要があるが、連続している必要はない。 0 (ゼロ) は、メッセージ ID 番号には無効です。
- メッセージ ID 番号の割り当ては、間に入るシンボル ID にも番号が付けられるように行う必要がある。 例えば、前の例のように、 1、2、OUTMSG、および 3 と行番号を付けた場合は、プログラムにエラーが含まれることになります。なぜなら、**mkcatdefs** コマンドもシンボル ID に番号を割り当て、数字の 3 を OUTMSG シンボル ID に割り当てているからです。

  注**:** シンボル ID は、メッセージ機能に固有です。 シンボル ID を使用した場合は、メッセージ・ソース・ファイルの移植性に影響が及びます。

メッセージ・ソース・ファイルへのコメントの追加

メッセージ・テキスト内を除いて、メッセージ・ソース・ファイルの任意の場所にコメントを組み込むことができます。 $ (ドル記号) の後には、スペースかタブ (ブランク) を少なくとも 1 つ置きます。

次に、コメントの例を示します。

```
$ This is a comment.
```

コメントは、メッセージ・ソース・ファイルから生成されたメッセージ・カタログには現れません。

コメントは、開発者がメッセージ・ソース・ファイルを維持し、変換プログラムが変換を行い、書き込み機能がメッセージの編集や文書化を行う過程で役立ちます。 **%s**、 **%c**、および **%d** など、変数が表す意味を識

別するときは、コメントを使用します。 例えば、変数が、ユーザー、ファイル、ディレクトリー、またはフラグのどれを参照するかを述べた注意書きを作成します。 古くなったメッセージを見分けるときにも、コメントを使用する必要があります。

分かりやすいように、コメント行は、メッセージ・カタログの最後ではなく、参照するメッセージのすぐ下に置くべきです。 セット全体のグローバル・コメントは、 **$set** ディレクティブのすぐ下に置くことができます。

次の行へのメッセージの継続

メッセージ番号の後のブランクに続くテキストは、行の終わりまで、すべてメッセージ・テキストとして組み込まれます。 メッセージ・テキストを次の行に継続する場合は、エスケープ文字の ¥ (バックスラッシュまたは円記号) を使用します。

¥ (バックスラッシュまたは円記号) は、次の例のように、行の最後の文字でなければなりません。

```
5 This is the text associated with ¥
message number 5.
```

これらの物理的な 2 行で、次の単一行のメッセージを定義します。

```
This is the text associated with message number 5.
```

注**:** メッセージ番号またはシンボル ID の後に複数のブランク文字を使用するのは、このメッセージ機能独自のものです。 複数のブランクを使用すると、メッセージ・ソース・ファイルの移植性に影響することがあります。

メッセージ・テキストへの特殊文字の組み込み

メッセージ・テキストに特殊文字を挿入するときは、 ¥ (バックスラッシュまたは円記号) を使用することができます。

これらの特殊文字を以下に示します。

特殊文字	説明
¥n	改行文字 (NL) の挿入。
¥t	水平タブ文字の挿入。
¥v	垂直タブ文字の挿入。
¥b	バックスペース文字の挿入。
¥r	復帰 (CR) 文字の挿入。
¥f	書式送り文字の挿入。
¥¥	¥ (バックスラッシュまたは円記号) の挿入。
¥*ddd*	有効な 8 進数字 *ddd* が表す 8 進値に関連付けられた 1 バイト文字を挿入します。 **注:** 1 から 3 桁の 8 進数字が指定できます。 しかし、8 進数字の後の文字も有効な 8 進数字であれば、先行ゼロを組み込む必要があります。 例えば、$ (ドル記号) の 8 進値は 44 です。 $5.00 を表示するには、¥445.00 ではなく ¥0445.00 を使用します。そうでないと、5 は 8 進値の一部として構文解析されます。
¥x*dd*	2 桁の有効な 16 進数字 *dd* で表される 16 進値に関連付けられた 1 バイト文字を挿入します。構文解析エラーを避けるには先行ゼロを組み込む必要があります (¥*ddd* の注を参照)。
¥x*dddd*	4 桁の有効な 16 進数字 *dddd* で表される 16 進値に関連付けられた 2 バイト文字を挿入します。構文解析エラーを避けるには先行ゼロを組み込む必要があります (¥*ddd* の注を参照)。

メッセージ・テキストを区切る文字の定義

メッセージ・テキストを区切る文字を定義するときは、メッセージ・ソース・ファイルの $quote ディレクティブを使用することができます。 この文字は ASCII 文字でなければなりません。

形式を次に示します。

```
$quote [character] [comment]
```

この指定文字は、メッセージ・テキストの前後に使用してください。次の例では、**$quote** ディレクティブで引用文字を _ (下線) に設定してから、それを引用符が入っている最後のメッセージの前で使用不可にします。

```
$quote _    Use an underscore to delimit message text
$set MSFAC         Message Facility - symbolic identifiers
SYM_FORM   _Symbolic identifiers can contain alphanumeric ¥
characters or the ¥_ (underscore character)¥n_
SYM_LEN    _Symbolic identifiers can be up to 65 ¥
characters long ¥n_
5              _You can mix symbolic identifiers and numbers ¥n_
$quote
MSG_H     Remember to include the _msg_h_ file in your program¥n
```

直前の例の最後の **$quote** ディレクティブは、下線文字を使用不可にします。

次の例では、 **$quote** ディレクティブは " (二重引用符) を引用文字として定義します。 引用文字は、メッセージ番号に続く最初の非ブランク文字でなければなりません。次に引用文字が発生した後のテキストは、すべて無視されます。

```
$quote "   Use a double quote to delimit message text
$set 10            Message Facility - Quote command messages
1   "Use the $quote directive to define a character ¥
¥n for delimiting message text"
2   "You can include the ¥"quote¥" character in a message ¥n ¥
by placing a ¥¥ in front of it"
3   You can include the "quote" character in a message ¥n ¥
by having another character as the first nonblank ¥
¥n character after the message ID number
$quote
4   You can disable the quote mechanism by ¥n ¥
using the $quote directive without a character ¥n¥
after it
```

上記の例は、引用文字をメッセージ・テキストに組み込むことができる 2 つの方法の説明です。

- 引用文字の前に ¥ (バックスラッシュまたは円記号) を置く。
- 引用文字以外の文字を、メッセージ番号に続く最初の非ブランク文字として使用する。 これで、この引用文字が使用不可になるのは、そのメッセージの場合のみです。

上記の例では、以下のことも示します。

- ¥ (バックスラッシュまたは円記号) は、複数行にまたがる引用メッセージを分割するときにも必要である。
- メッセージ内で ¥ (バックスラッシュまたは円記号) を表示するには、その前にもう 1 個の ¥ (バックスラッシュまたは円記号) を入れる。
- **¥n** を使用して、改行文字 (NL) でメッセージのフォーマット設定ができる。
- 文字引数を指定せずに **$quote** ディレクティブを使用すると、引用メカニズムが使用不可になる。

メッセージ・セット番号とメッセージ ID 番号の割り当て

すべてのメッセージ・セットに、セット番号またはシンボル ID が必要です。

ソース・ファイルに **$set** ディレクティブを使用して、メッセージのグループに、番号または ID を指定します。

$set *n* [*comment*]

メッセージ・セット番号は *n* の値 (1 から NL_SETMAX の範囲内の数字) によって指定します。 数値ではなく、シンボル ID を使用することができます。 次に **$set** ディレクティブが現れるまで、 **$set** ディレクティブに続くすべてのメッセージが、そのセット番号に割り当てられます。デフォルトのセット番号は 1 です。セット番号は昇順に割り当てられなければなりませんが、連続している必要はありません。 番号をスキップすると、空のセットが作られます。 しかし、数列に大きなギャップがあると、効率やパフォーマンスが減少することがあります。 さらに、カタログで複数のセット番号を使用すると、パフォーマンスは向上しません。

次のように、 **$set** ディレクティブにはコメントを組み込むこともできます。

```
$set 10  Communication Error Messages

$set OUTMSGS  Output Error Messages
```

AIX メッセージ・セットの多くには、**MS**_*PROG* 形式のシンボル ID が付いています。ここで、**MS** はメッセージ・セットを表し、*PROG* はそのメッセージ・セットに関連するプログラムまたはユーティリティーの名前です。 例えば、次のようになります。

```
$set MS_WC  Message Set for the wc Utility

$set MS_XLC1  Message Set 1 for the C For AIX compiler

$set MS_XLC2  Message Set 2 for the C For AIX compiler
```

カタログからのメッセージの除去

このセクションでは、**$delset** ディレクティブを使用してカタログからメッセージを除去する方法について説明します。

$delset ディレクティブは、指定したセットに属するすべてのメッセージを、既存のカタログから除去します。

$delset *n* [*comment*

メッセージ・セットは、*n* によって指定されます。 **$delset** ディレクティブは、同じソース・ファイルにある **$set** ディレクティブに対して正しいセット番号順に置く必要があります。 **$delset** ディレクティブには、コメントを組み込むこともできます。

メッセージ・テキストの長さ

このセクションでは、メッセージ・テキストの表示の最大長について説明します。

$len ディレクティブは、メッセージ・テキストを表示する最大長を設定します。

$len [*n* [*comment*]]

n を指定しないか、あるいは **$len** ディレクティブが組み込まれない場合、メッセージ・テキストの表示は **NL_TEXTMAX** 値に設定されます。メッセージ・テキストの表示の長さとは、メッセージに使用できる最大バイト数です。 後続の任意の **$len** ディレクティブの指定で、前の指定は指定変更されます。 n の値は、 **NL_TEXTMAX** 値を超えることはできません。

メッセージ・テキストの内容

可能なときは常に、発生したこと、および状況の修復のためにできることをユーザーに正確に伝えます。

次の例で、原因とリカバリーの情報によってメッセージを改善する方法を示します。

```
Original  Message:   Bad arg

Revised Message:   Specify year as a value between 1 and 9999.
```

「Bad arg」というメッセージは、あまりユーザーには役立ちません。これに対し、「Do not specify more than 2 files on the command line」であれば、コマンドが機能するために行うべきことがユーザーに正確に伝わります。 同様に、「Line too long」というメッセージでは、ユーザーのリカバリー情報にはなりません。 「Line cannot exceed 20 characters」のメッセージは、欠けている情報を与えてくれます。

メッセージ・ソース・ファイルの例

このトピックでは、メッセージ・ソース・ファイルの例を示します。

次の例では、メッセージ・ソース・ファイルは、メッセージ ID 番号およびメッセージ・セット番号に数字を使用しています。

```
$ This is a message source file sample.
$ Define the Quote Character.
$quote "
$set 1 This is the set 1 of messages.
1 "The specified file does not have read permission on¥n"
2 "The %1$s file and the %2$s file are same¥n"
3 "Hello world!¥n"
$Define the quote character
$quote '
$set 2 This is the set 2 of messages
1       'fieldef: Cannot open %1$s ¥n'
2       'Hello world¥n'
```

次の例では、メッセージ・ソース・ファイルは、メッセージ ID 番号およびメッセージ・セット番号にシンボル ID を使用しています。

```
$ This is a message source file sample.
$ Define the Quote Character.
$quote "
$set MS_SET1 This is the set 1 of messages.
MSG_1   "The specified file does not have read permission on¥n"
MSG_2   "The %1$s file and the %2$s file are same¥n"
MSG_3   "Hello world¥n"
$Define the quote character
$quote
$set 2 This is the set 2 of messages.
$EMSG_1 'fieldef: Cannot open %1$s¥n'
$EMSG_2 'Hello world!¥n'
```

次の例は、シンボル ID を使用すると、メッセージの指定が分かりやすくなることを示しています。

```
catgets(cd, 1, 1, "default message")

catgets(cd, MS_SET1, MSG_1, "default message")
```

メッセージ・カタログの作成

メッセージ機能には、外部化されたメッセージ・カタログにあるプログラム・メッセージを検索し、表示するためのコマンドやサブルーチンがあります。 プログラマーは、アプリケーション・メッセージが入って

いるメッセージ・ソース・ファイルを作成し、それをメッセージ・カタログに変換します。 メッセージ・ソース・ファイルを他の言語に変換してから、ファイルをメッセージ・カタログに変換するには、プログラムの変更や再コンパイルは必要ありません。

メッセージ・カタログを作成するには、メッセージ機能の **gencat** コマンドによって、完成しているメッセージ・ソース・ファイルを処理します。 このコマンドは、次の方法で使用することができます。

- **gencat** コマンドを使用して、セット番号、メッセージ ID 番号、およびメッセージ・テキストが入っているメッセージ・ソース・ファイルを処理する。 シンボル ID が入っているメッセージ・ソース・ファイルは、 gencat コマンドで直接処理することはできません。 次の例は、**x.msg** メッセージ・ソース・ファイルの情報を用いてカタログ・ファイルを生成するものです。

  ```
  gencat x.cat x.msg
  ```

- **mkcatdefs** コマンドを使用して、シンボル ID が入っているメッセージ・ソース・ファイルをプリプロセスする。この結果、ファイルは **gencat** コマンドにパイピングされます。 **mkcatdefs** コマンドは、定義ステートメントを含む *SymbolName*_**msg.h** ファイルを作成します。 これらのステートメントは、シンボル ID と、 **mkcatdefs** コマンドによって割り当てられた、セット番号およびメッセージ ID 番号を同等にします。 *SymbolName*_**msg.h** ファイルは、これらのシンボル ID を用いてプログラムに組み込む必要があります。 **mkcatdefs** コマンドは、AIX に固有です。 次の例は、**x.msg** メッセージ・ソース・ファイル内の情報を用いて、 **x_msg.h** ヘッダー・ファイルを作成します。

  ```
  mkcatdefs x x.msg
  ```

- **runcat** コマンドを使用して、シンボル ID が入っているソース・ファイルを自動的に処理する。 **runcat** コマンドは、**mkcatdefs** コマンドを起動し、その出力を **gencat** コマンドにパイピングします。 **runcat** コマンドは、AIX に固有です。 次の例は、`x.msg` メッセージ・ソース・ファイル内の情報を用いて、 **x_msg.h** ヘッダー・ファイルと **X.cat** カタログ・ファイルを生成します。

  ```
  runcat x x.msg
  ```

上記の例は、次の例と同じです。

```
mkcatdefs x x.msg | gencat x.cat
```

CatalogFile パラメーターで名前を指定されているメッセージ・カタログが存在する場合、 **gencat** コマンドは、メッセージ・ソース・ファイルのステートメントに従ってそのカタログを変更します。 メッセージ・カタログが存在しない場合、 **gencat** コマンドは *CatalogFile* パラメーターに指定されている名前でカタログ・ファイルを作成します。

ユーザーは任意の数のメッセージ・テキスト・ソース・ファイルを指定することができます。 指定した順序で、複数のファイルが処理されます。 後続の各ソース・ファイルは、カタログを変更します。 ソース・ファイルを指定しない場合、 **gencat** コマンドは標準入力からメッセージ・ソース・データを受け入れます。

名詞または名詞句 (例: クライアント証明書)

このトピックでは、カタログのサイズ変更と、サイズの定義に使用されるマクロについて説明します。

メッセージ・カタログのサイズは、事実上任意です。 カタログ内のセット、カタログ内のメッセージ、およびメッセージ内のバイトの各最大数は、以下のマクロによって **/usr/include/limits.h** ファイルで定義されます。

マクロ	説明
NL_SETMAX	**$set** ディレクティブによって指定することができるセット番号の最大数を指定する。 **NL_SETMAX** 制限を超えると、 **gencat** コマンドはエラー・メッセージを出して、メッセージ・カタログの作成も、更新も行いません。
NL_MSGMAX	システムが容認するメッセージ ID 番号の最大数を指定する。 **NL_MSGMAX** 制限を超えると、 **gencat** コマンドはエラー・メッセージを出して、メッセージ・カタログの作成も、更新も行いません。
NL_TEXTMAX	メッセージに含めることができる最大バイト数を指定する。 **NL_TEXTMAX** 制限を超えると、 **gencat** コマンドはエラー・メッセージを出して、メッセージ・カタログの作成も、更新も行いません。

例

このセクションでは、ソース・ファイルからメッセージ・カタログを作成する方法に関するオプションについて説明します。

多くの場合、「これは何ですか?」という質問に答えて、定義で始まります。

- この例は、メッセージ識別番号を含むソース・ファイルからメッセージ・カタログを作成する方法を示します。 次に示すのは、 **hello.msg** メッセージ・ソース・ファイルのテキストです。

  ```
  $ file: hello.msg
  $set 1    prompts
  1 Please, enter your name.
  2 Hello, %s ¥n
  $ end of file: hello.msg
  ```

  hello.cat メッセージ・カタログを **hello.msg** ソース・ファイルから作成するには、次のように入力します。

  ```
  gencat hello.cat hello.msg
  ```

- この例は、シンボル参照を用いるソース・ファイルからメッセージ・カタログを作成する方法を示します。 次に示すのは、メッセージ・セットおよびメッセージへのシンボル参照が入っている **hello.msg** メッセージ・ソース・ファイルのテキストです。

  ```
  $ file: hello.msg
  $quote "
  $set PROMPTS
  PLEASE  "Please, enter your name."
  HELLO   "Hello, %s ¥n"
  $ end of file: hello.msg
  ```

  hello.msg および **msgerrs** メッセージ・ソース・ファイルを処理するには、次のように入力します。

  ```
  runcat hello hello.msg
  runcat msgerrs msgerrs.msg /usr/lib/nls/msg/$LANG/msgerrs.cat
  ```

  runcat コマンドは、**mkcatdefs** および **gencat** コマンドを起動します。 **runcat** コマンドの最初の呼び出しで、 **hello.msg** ソース・ファイルを取り込み、 2 番目のパラメーター `hello` を使用して **hello.cat** メッセージ・カタログと **hello_msg.h** 定義ファイルを作成します。

  hello_msg.h 定義ファイルには、メッセージ・カタログのシンボル名と、メッセージおよびセットのシンボル ID が入っています。 **hello.cat** メッセージ・カタログのシンボル名は `MF_HELLO` です。この名前は、**mkcatdefs** コマンドによって自動的に作成されます。

  `runcat` コマンドの 2 回目の呼び出しで、msgerrs.msg ソース・ファイルを取り込み、最初のパラメーター `msgerrs` を使用して **msgerrs_msg.h** 定義ファイルを作成します。

3 番目のパラメーター /usr/lib/nls/msg/$LANG/msgerrs.cat が存在するので、 **runcat** コマンドはカタログ・ファイル名にこのパラメーターを使用します。 このパラメーターは、 **runcat** コマンドがファイルを収めなければならない場所を正確に指定する、絶対パス名です。 msgerrs.cat カタログのシンボル名は MF_MSGERRS です。

アプリケーション・プログラムの外でのメッセージの表示

以下のコマンドを用いると、メッセージをアプリケーション・プログラムの外で表示することができます。これらのコマンドは AIX に固有です。

コマンド	説明
dspcat	指定されたメッセージ・カタログに入っているメッセージを表示する。 次の例は、**x.cat** メッセージ・ソース・ファイル内にあるメッセージを表示します。 `dspcat x.cat`
dspmsg	メッセージ・カタログから単一のメッセージを表示する。 次の例は、ID 番号が 1、セット番号が 2 の **x.cat** メッセージ・ソース・ファイル内にあるメッセージを表示します。 `dspmsg x.cat -s 2 1` メッセージをメッセージ・カタログから入手しなければならないときは、シェル・スクリプトの **dspmsg** コマンドを使用することができます。

アプリケーション・プログラムを用いたメッセージの表示

このトピックでは、ご使用のアプリケーション・プログラミングに含める必要のある項目について説明します。

メッセージ機能を用いてプログラミングを行うときは、アプリケーション・プログラムに以下の項目を組み込む必要があります。

- メッセージ・ソース・ファイルでシンボル ID を使用した場合は **mkcatdefs** または **runcat** コマンドによって作成された *CatalogFile*_**msg.h** 定義ファイル。メッセージ・ソース・ファイルでシンボル ID を使用しなかった場合は **limits.h** および **nl_types.h** ファイル。
- ロケール環境を初期化する呼び出し
- カタログをオープンする呼び出し
- メッセージを読み取る呼び出し
- メッセージを表示する呼び出し
- カタログをクローズする呼び出し

以下のサブルーチンは、メッセージ機能によるプログラム・メッセージの表示に必要なサービスを備えています。

サブルーチン	説明
setlocale	ロケールを設定する。 優先メッセージ・カタログ言語用の **setlocale** サブルーチンの呼び出しの中で、**LC_ALL** 環境変数を指定する。
catopen	指定したメッセージ・カタログをオープンし、カタログからメッセージを検索するときに使用する、カタログ記述子を戻す。
catgets	**catopen** サブルーチンが正常に呼び出された後、カタログからメッセージを検索する。
printf	stdout (標準出力) ストリームの変換、フォーマット設定、およびそれへの書き出しを行う。

サブルーチン	説明
catclose	指定したメッセージ・カタログをクローズする。

次の C プログラム hello では、 **catopen** サブルーチンによる hello.cat カタログのオープン、 **catgets** サブルーチンによるカタログからのメッセージの検索、 **printf** サブルーチンによるメッセージの表示、および **catclose** サブルーチンによるカタログのクローズの例を示しています。

```
/* program: hello */
#include <nl_types.h>
#include <locale.h>
nl_catd catd;
main()
{
/*  initialize the locale  */
setlocale (LC_ALL, "");
/* open the catalog */
catd=catopen("hello.cat",NL_CAT_LOCALE);
printf(catgets(catd,1,1,"Hello World!"));
catclose(catd);                     /* close the catalog */
exit(0);
}
```

上記の例では、**catopen** サブルーチンは、 **hello.cat** メッセージ・カタログをファイル名だけで参照しています。 したがって、**NLSPATH** 環境変数が正しく設定されていることを確認する必要があります。 メッセージ・カタログが **catopen** サブルーチンによって正常にオープンされた場合には、 **catgets** サブルーチンは、 hello.cat カタログで指定されているメッセージを指すポインターを戻します。 メッセージ・カタログが見つからないか、カタログにメッセージが存在しない場合、 **catgets** サブルーチンは、デフォルト・ストリング Hello World! を戻します。

NLSPATH 環境変数の理解

NLSPATH 環境変数で、メッセージ・カタログを検索するディレクトリーを指定します。

catopen サブルーチンは、メッセージ・カタログを見つけてオープンするためのコールで指定された順序に基づいて、これらのディレクトリーを検索します。メッセージ・カタログが検出されない場合、メッセージ検出ルーチンは、プログラム提供のデフォルト・メッセージを戻します。 **NLSPATH** デフォルト・パスについては **/etc/environment** ファイルを参照してください。

プログラム提供のデフォルト・メッセージの検索

メッセージ検索ルーチンは、なんらかの理由で希望するメッセージが検索できない場合に、すべてのプログラム提供のデフォルト・メッセージ・テキストを戻します。 プログラム提供のデフォルト・メッセージは、一般に、テキスト内にメッセージ番号のない簡単な 1 行のメッセージです。 このようなデフォルト・メッセージを使用する場合は、 C ロケールに **LC_MESSAGES** カテゴリーを設定するか、あるいは **NLSPATH** 環境変数の設定を解除します。 **LC_ALL**、 **LC_MESSAGES**、あるいは **LANG** 環境変数のいずれも設定されていない場合、 **LC_MESSAGES** カテゴリーがデフォルトの C ロケールになります。

言語階層の設定

マルチリンガル・ユーザーは、メッセージ・テキストの言語階層を指定することができます。

システム・デフォルト用、または個々のユーザー用に言語階層を設定するときは、 5 ページの『言語環境の変更』または SMIT を参照してください。 SMIT を用いて言語階層を設定する場合は、コマンド・ラインに SMIT 高速パス smit mlang を入力します。

「言語階層の変更/表示」を選択します。

または

コマンド・ラインで、以下のように入力します。

```
smit
```

「システム環境」を選択します。

「言語環境の管理 **(Manage Language Environment®)**」を選択します。

「言語階層の変更/表示」を選択します。

カタログからのメッセージ検索の例

この例は 3 つの部分に分けることができます。すなわち、メッセージ・ソース・ファイル、メッセージ・カタログの生成に使用するコマンド、およびメッセージ・カタログを使用するプログラムの例です。

1. 次の例は、**example.msg** メッセージ・ソース・ファイルを示します。

   ```
   $quote "
   $ every message catalog should have a beginning set number.
   $set MS_SET1
   MSG1 "Hello world¥n"
   MSG2 "Good Morning¥n"
   ERRMSG1 "example: 1000.220 Read permission is denied for the file
   %s.¥n"
   $set MS_SET2
   MSG3 "Howdy¥n"
   ```

2. 次のコマンドは、 **example.msg** メッセージ・ソース・ファイルを使用して、現行ディレクトリー内に **example.h** ヘッダー・ファイルと **example.cat** カタログ・ファイルを生成します。

   ```
   runcat example example.msg
   ```

3. 次のプログラム例は、 **example.h** ヘッダー・ファイルを使用して、 **example.cat** カタログ・ファイルにアクセスしています。

   ```
   #include <locale.h>
   #include <nl_types.h>
   #include "example_msg.h" /*contains definitions for symbolic
                              identifiers*/
   main()
   {
           nl_catd catd;
           int error;

           (void)setlocale(LC_ALL, "");

           catd = catopen(MF_EXAMPLE, NL_CAT_LOCALE);
           /*
           ** Get the message number 1 from the first set.
           */
           printf( catgets(catd,MS_SET1,MSG1,"Hello world¥n") );

           /*
           ** Get the message number 1 from the second set.
           */
           printf( catgets(catd, MS_SET2, MSG3,"Howdy¥n") );
           /*
           ** Display an error message.
           */
           printf( catgets(catd, MS_SET1, ERRMSG1,"example: 100.220
                   Permission is denied to read the file %s.¥n") ,
                   filename);
           catclose(catd);
   }
   ```

メッセージの作成

このセクションでは、メッセージを意義のあるものにし、簡潔にするのに役立つヒントを記載します。

以下のヒントは、メッセージを分かりやすくし、簡潔にするのに助けになります。

- パネル上で表示するメッセージを含む、すべてのメッセージのグローバリゼーションに関して計画を立てる。
- 変換されたメッセージの表示に、十分なスペースを用意する。変換されたメッセージは、しばしば元のメッセージ・テキストより大きな表示列を占有します。 一般には、変換されたメッセージに約 20% から 30% 増しのスペースを用意しますが、場合によっては、変換されたメッセージに 100% 増しのスペースが必要になる場合があります。
- メッセージ・カタログを使用して、ユーザー・メッセージおよびエラー・メッセージを外部化する。 X アプリケーションは、リソース・ファイルを用いて、ロケールごとにメッセージを外部化します。
- デフォルト・メッセージを用意する。
- メッセージ・ソース・ファイル内の各メッセージを、完全なエンティティーにする。部分を連結してメッセージを作成すると、変換が困難になります。
- メッセージ・テキストの最大表示長を制御するには、メッセージ・ソース・ファイルの **$len** ディレクティブを使用する。 (**$len** ディレクティブは、メッセージ機能に固有です。)
- セット番号およびメッセージ番号の指定には、シンボル ID を使用する。 プログラムは、実際の番号ではなく、シンボル ID によって、セット番号およびメッセージ番号を参照する必要があります。 (シンボル ID の使用は、メッセージ機能に固有です。)
- **%s** 変数に番号を付けることによって、文節の再配列を容易にする。これで、変換プログラムは必要な場合に文節を再配列することができます。 例えば、プログラムが英語メッセージ「`The file %s is referenced in %s`」を表示する必要がある場合、プログラムは、以下のように 2 つのストリングを指定することができます。

  ```
  printf(message_pointer, name1, name2)
  ```

  英語メッセージは、以下のように **%s** 変数に番号を付けます。

  ```
  The file %1$s is referenced in %2$s¥n
  ```

  変換されたこのメッセージの同じ内容は、次のようになります。

  ```
  %2$s contains a reference to file %1$s¥n
  ```
- エラー・メッセージを入手するのに、 **sys_errlist[errno]** を使用しない。これは、メッセージ外部化の目的を損ないます。 **sys_errlist[]** は、英語でのみ提供されるエラー・メッセージの配列です。 **strerror**(*errno*) を使用します。この場合、メッセージはカタログから入手します。
- エラー・メッセージを入手するのに、 **sys_siglist[signo]** を使用しない。これは、メッセージ外部化の目的を損ないます。 **sys_siglist[]** は、英語でのみ提供されるエラー・メッセージの配列です。 **psignal()** を使用します。この場合、メッセージはカタログから入手します。
- メッセージ・コメント機能を用いて、メッセージの保守および変換を補助する。
- 一般に、各コマンドもしくはユーティリティーに適用するメッセージの、個別のメッセージ・ソース・ファイルやカタログを作成する。

メッセージ内のコマンド構文の記述

このセクションでは、使用ステートメント内のコマンド構文を示します。

- 使用ステートメントのコマンド構文を表示する。 **rm** コマンドの使用ステートメントとして次の例が考えられます。

```
Usage: rm [-firRe] [--] File ...
```

- 使用ステートメント・メッセージ内の File、 Directory、 String、および Number などの単語の最初の文字を大文字にする。
- コマンド・ラインのパラメーターは省略しない。 例えば、Num を Number と入力したほうが、簡単に変換できます。
- 使用ステートメント・メッセージでは、以下の区切り文字のみを使用する。

区切り文字	説明
[]	オプション・パラメーターを囲む。
{}	いずれか 1 つが必要である、複数のパラメーターを囲む。
|	両方は選択できないパラメーターを分離する。 例えば、**[a|b]** は、 a か b を選択できるか、 a と b を選択できないことを示し、 {a|b} は a か b を選択しなければならないことを示す。
...	コマンド・ラインで繰り返すことができるパラメーターの後に続ける。省略符号の前にスペースがあることに注意してください。
-	標準入力を示す。

- 唯一の選択である必須パラメーターには、区切り文字を使用しない。 例えば、次のようになります。

```
banner String
```

- コマンド・ラインで分離する必要があるフラグ間に間隔文字を入れる。 例えば、次のようになります。

```
unget [-n] [-rSID] [-s] {File|-}
```

- コマンド・ラインで一緒に使用できるフラグは分離しない。 例えば、次のようになります。

```
wc [-cwl] {File ...|-}
```

- コマンド・ラインのフラグの順序に差がないときは、フラグをアルファベット順に置く。 大文字フラグの前に小文字フラグを置く。 例えば、次のようになります。

```
get -aAijlmM
```

- 使用ステートメント・メッセージの行を終了する場所を決めるときは、十分に考慮する。 次の例で、長い使用ステートメント・メッセージを示します。

```
Usage: get [-e|-k] [-c Cutoff] [-i List] [-r SID] [-w String]
 [-x List] [-b] [-gmnpst] ...
```

必要な場合は、2 行目に使用情報を継続してください。 例えば、次のようになります。

```
Usage: get [-e|-k] [-c Cutoff] [-i List] [-r SID] [-w String]
           [-x List] [-b] [-gmnpst] [-l[p]] File ...
```

メッセージの作成スタイル

明瞭な書き方は、メッセージの変換を助けます。 メッセージの作成スタイルに関する以下の指針には、用語、句読点、叙法、態、時制、大文字、形式、およびその他の使用上の問題点が含まれます。

- 簡潔なメッセージを作成する。 望ましいのは、1 つの文のメッセージです。
- 完全文形式を使用する。
- あいまいさを除く必要があるときは、冠詞 (a、an、the) を追加する。
- 文の最初のワードを大文字ではじめ、文の終わりにピリオドを使用する。
- 現在形を使用する。 メッセージには未来形は使用しないでください。 例えば、次の文を使用します。

```
The cal command displays a calendar.
```

次の文は使用しません。

```
The cal command will display a calendar.
```

- メッセージでは一人称 (I または we) は使用しない。

- ヘルプ・テキストや対話式テキスト以外では、二人称 (you) の使用を避ける。
- 能動態を使用する。 次の例は、受動態で書かれたメッセージを、どのように能動態のメッセージに戻せるかを示しています。

  Passive: Month and year must be entered as numbers.
  Active: Enter month and year as numbers.
- 命令法 (コマンド句) および能動動詞 (specify、use、check、choose、および wait などの) を使用する。
- メッセージを肯定語調で述べる。 以下に、否定的メッセージを肯定的にした例を示します。

  Negative: Don't use the f option more than once.
  Positive: Use the -f flag only once.
- ワードは、辞書にある文法上のカテゴリーでのみ使用する。名詞でのみ表されるワードは、動詞としては使用しないでください。 例えば、「問題をソリューション *(solution)* する」や「システムをアーキテクト *(architect)* する」などは使用しないでください。
- 接頭部も接尾部も使用しない。 変換プログラムは、re-、un-、in-、あるいは non- で始まるワードの意味が分からない場合があり、接頭部もしくは接尾部を使用したメッセージの変換が、意図する意味にならないことがあります。 この規則に例外が生ずるとすれば、接頭部が、使用頻度の高いワードの必須の要素である場合です。 例えば、previous と premature のワードは受け入れられますが、 nonexistent は受け入れられません。
- 単数または複数を示す括弧は使用しない。例えば、error(s) の括弧は変換できません。 単数および複数を示す必要がある場合は、*error or errors* と書きます。 ワードの単数または複数の必要の有無によって、異なるメッセージが出されるように、コードを変えることもできます。
- 短縮形は使用しない。
- 引用符は、単一引用符も二重引用符も使用しない。例えば、**%s**、 **%c**、および **%d** などの変数の周囲、またはコマンドの周囲には引用符を使用しないでください。 ユーザーが、引用符を文字通りに解釈する可能性があります。
- 行の終わりのワードにハイフンを付けない。
- メッセージでは、標準の強調表示指針を適用せず、頭文字または大文字を他の強調表示方法で置き換えることはしない。 (標準の強調表示には、コマンド、サブルーチン、およびファイル用の太字、変数やパラメーター用のイタリック、例や表示テキスト用のタイプライター書体または Courier フォントのような指針が含まれます。)
- and/or の構造は使用しない。 このような構造は、他の言語には存在しません。 通常、両方を行う必要がないことを示す場合は、 or のほうが適切な表現です。
- 24 時間クロックを使用する。 時刻の指定に a.m. または p.m. を使用しないでください。例えば、1:00 p.m. は 1300 と書きます。
- 頭字語は避ける。 頭字語の使用は、完全なスペルの用語より読者にとって分かりやすい場合に限ります。 頭字語を複数形にする場合は、アポストロフィを付けずに小文字の s を追加します。頭字語は、使用の前に、商標でないことを確認してください。
- メッセージを文節から作成しない。 プログラム内でフラグまたは他の手段を使用して情報を伝え、完全なメッセージが適切な時点で出されるようにします。
- ハードコーディングされたテキストを、メッセージ内の **%s** ストリングの変数として使用しない。
- メッセージの最後の行を **¥n** (改行を示す) で終了する。 これは、1 行メッセージにも該当します。
- 2 番目以降の行を **¥t** (タブを示す) で始める。
- 他のすべての行を **¥n¥** (改行を示す) で終了する。

- 必要ならば、ワード境界で強制的に改行し、メッセージ・ストリングの表示を受け入れられるようにする。 **printf** サブルーチンは、しばしば、メッセージ・テキストの表示に使用されますが、ワード境界を無視し、ときにはワードを中央で分割してテキストを必要に応じて折り返します。
- なんらかの理由で、メッセージを改行文字で終えてはならない場合は、作成者にその趣旨のコメントを残す。
- 各メッセージの前に、メッセージを呼び出したコマンドの名前を付け、その後にコロンを付ける。 次の例は、コマンド名が入っているメッセージです。

  ```
  OPIE "my_example: Opening the file."
  ```

- システムが過負荷にならない限り、ユーザーに**後で再試行してください**と指示しない。 再試行の必要は、メッセージから明白なはずです。
- コマンド・ラインのテキストを記述するときは、 parameter (パラメーター) という語、数値データを示す場合は、value (値) という語、パラメーターとともにコマンドを記述する場合は、 command string (コマンド・ストリング) という語を使用する。
- 値の 3 桁ごとの桁区切りのコンマは使用しない。 `1,000` ではなく 1000 を使用してください。
- メッセージを * (アスタリスク) で強調する必要がある場合には、メッセージの先頭と終わりに 2 つずつのアスタリスクを付ける。 例えば、次のようになります。

  ```
  ** Total **
  ```

- log in (ログインする) や log off (ログオフする) という語は動詞として使用する。例えば、次のようになります。

  Incorrect: `Choose the appropriate method for system log in.`
  Correct: `Choose the appropriate method to log in to the system.`

- user name (ユーザー名)、group name (グループ名)、および login (ログイン) という語は名詞として使用する。例えば、次のようになります。

  ```
  The user is sam.
  The group name is staff.
  The login directory is /u/sam.
  ```

- superuser (スーパーユーザー) という語は使用しない。 root ユーザーにすべての特権があるとは限らないことに注意してください。
- 適用できるならば、以下の発生頻度の高い標準のメッセージを使用する。

望ましい標準メッセージ	あまり望ましくないメッセージ
Cannot find or open the file.	Can't open filename.
Cannot find or access the file.	Can't access.
The syntax of a parameter is not valid.	Syntax error.

各国固有のデータ処理

各国固有のデータ処理がプログラムの一部である場合があり、このようなプログラムは、多様なロケールに対してそれぞれに対応するデータを提供する場合があります。さらに、プログラムは、言語および各国ごとに文字データを処理するのに、別々のアルゴリズムを使用する場合があります。

例えば、ワードの開始と終了の認識や、2 行にまたがるワードのハイフンの付け方は、ロケールによって異なっています。 このような機能を取り扱うプログラムでは、実行時の現行ロケールの設定に基づいて該当のテーブルもしくはアルゴリズムにアクセスする必要があります。 このようなプログラムは、以下のように処理することができます。

- アルゴリズムおよびテーブルをすべてコンパイルし、それをプログラムとともにロードする。

この方法を採用する場合には、アルゴリズムやテーブルの追加もしくは変更は困難になります。 新規アルゴリズムもしくはテーブルが追加されるたびに、必ずプログラム全体をリンクし直す必要があるからです。

- ロケール固有のアルゴリズムおよびテーブルをファイル内に保持し、それを実行時に現行ロケールの設定に従ってロードする。

  この方法を採用する場合には、アルゴリズムやテーブルの追加および変更が容易になります。 ただし、アルゴリズムをロードするために定義された標準の方法はありません。 AIX の場合は、**load** サブルーチンを用いてこれを行うことができますが、**load** サブルーチンを使用するプログラムは、他のシステムに移植できない場合があります。

各国固有のテーブル

各国固有のデータを、現行ロケールの設定に基づいてテーブルにアクセスすることによって処理できるのであれば、これは、標準ファイル入出力サブルーチン (**fopen**、**fread**、**open**、**read** など) を用いて行うことができます。 この種のテーブルは、 **/usr/lpp/***Name* で定義されたディレクトリーに提供する必要があります。ここで *Name* とは、該当するロケール名のもとの特定アプリケーションの名前です。

標準パス接頭部
: **/usr/lpp/***Name* (AIX 固有のパス名)

各国固有のディレクトリー
: テーブルを記述する、適切なカテゴリーの現行ロケールを入手する。 これを上記の接頭部に連結します。

アクセス
: 標準ファイル・アクセス・サブルーチン (**fopen**、**fread**、など) を適切に使用する。

各国固有のアルゴリズム

各国固有のアルゴリズムは、**/usr/lpp/***Name***/%L** ディレクトリーに存在します。ここで、**%L** は、該当するカテゴリーの現行ロケール設定を表します。

load サブルーチンを用いて、オブジェクト・モジュールからプログラム固有のアルゴリズムにアクセスします。

標準パス接頭部
: **/usr/lpp/***Name*

各国固有のディレクトリー
: 該当するカテゴリーの現行ロケールを入手する。 これを上記の接頭部に連結します。

メソッド
: これにメソッド名を連結する。

アプリケーションのためにアラビア語テキスト用の各国固有モジュールをロードする例

このセクションでは、アプリケーションのために、アラビア語テキスト用の各国固有モジュールをロードすることについて説明します。

ヘッダー・ファイル

このセクションでは、**method.h** 組み込みファイルにある 1 つの構造体について説明します。

methods.h 組み込みファイルには、次のように 1 つの構造体があります。

```
struct Methods {
        int     version;
        char    *(*hyphen)();
        char    *(*wordbegin)();
        char    *(*wordend)();
} ;
```

主プログラム

この例では、プログラム名を **textpr** とします。

主プログラムは、ロードするモジュールを決めて、それを呼び出します。ロード・オブジェクトのパス名を指定するために textpr.h 組み込みファイルを使用することに注意してください。 このように、パス名は、システムに固有であり、簡単に変更することができます。

```
#include    <stdio.h&gt;
#include    <errno.h&gt;
#include    "methods.h"
#include    "textpr.h"    /* contains the pathname where
                             the load object can be found */

extern     int   errno;

main()
{
     char libpath[PATH_MAX];&rbl;&rbl;&rbl;/* stores the full pathname of the
                                  load object */
      char *prefix_path=PREFIX_PATH;        /* from textpr.h */
      char *method=METHOD;                  /* from textpr.h */
      int (*func)();
      char *path;
      /* Methods */
      int   ver;
      char  *p;
      struct Methods *md;


      setlocale(LC_ALL, "");

      path = setlocale(LC_CTYPE, 0);    /* obtain the locale
                                           for LC_CTYPE category */
      /* Construct the full pathname for the */
      /* object to be loaded                 */
      strcpy(libpath, prefix_path);
      strcat(libpath, path);
      strcat(libpath, "/");
      strcat(libpath, method);

      func = load(conv, 1, libpath);        /* load the object */
      if(func==NULL){
            strerror(errno);
            exit(1);
      }
      /* invoke the loaded module ");
      md =(struct Methods *) func();       /* Obtain the methods
                                              structure */
      ver = md-&gt;version;
      /* Invoke the methods as needed */
```

```
        p = (md-&gt;hyphen)();
        p = (md-&gt;wordbegin)();
        p = (md-&gt;wordend)();
}
```

メソッド

このセクションには、各国固有のアルゴリズムが含まれています。

この例では、アラビア語メソッドが提供されています。 method.c プログラムが続きます。

```
#include "methods.h"

char *Arabic_hyphen(char *);
char *Arabic_wordbegin(char *);
char *Arabic_wordend(char *);

static struct Methods ArabicMethods= {
        1,
        Arabic_hyphen,
        Arabic_wordbegin,
        Arabic_wordend
} ;

struct Methods *start_methods()
{
        /* startup methods */
        return ( &ArabicMethods);
}

char *Arabic_hyphen(char *string)
{
        /* Arabic hyphen */
        return( string );
}
char *Arabic_wordbegin(char *string)
{
        /*Arabic word begin */);
        return( string );
}
char *Arabic_wordend(char *string)
{
        /* Arabic word end */;
        return( string);
}
```

組み込みファイル

textpr 組み込みファイルには、ロードされるモジュールのパス名が入ります。

```
#define PREFIX_PATH "/usr/lpp/textpr"
                    /* This is an AIX-specific pathname */
```

レイアウト (両方向テキストと文字成形) の概要

両方向 (BIDI) テキストが発生するのは、方向の異なるテキストが一緒に現れる場合です。 例えば、英語のテキストは左から右に読みます。 アラビア語とヘブライ語のテキストは右から左に読みます。 同じ行に英語とヘブライ語のテキストが現れれば、このテキストは両方向です。

両方向テキストと字形に関する詳細、および入手可能な資料のリストは、以下の Web サイトを参照してください。

http://www.opengroup.org

両方向テキストは、以下のガイドラインに従って書きます。

- アラビア語とヘブライ語のテキストは右から左に書きます。 (ストリングとは、アルファベットと数字の環境で順序付けることを目的とするワードと考えられています。)
- 数および英語の句は、左から右に書かれます。
- 数字とその句読記号は、左から右に書かれます。

両方向のスクリプトは、右から左、上から下に読まれます。

組み込みテキストが 1 行内に入れられる場合は、テキストは左から右に書かれ、両方向テキストに組み込まれます。 しかし、その組み込みテキストが複数行に分割されている場合は、上から下に読めるように、左から右の部分に正しい順序が保たれる必要があります。

例えば、1 行内に含まれた左から右のテキストに組み込まれた、右から左のテキストは、次のように書かれます。

```
THERE IS txet lanoitceridib deddebme IN THIS SENTENCE.
```

左から右のテキストに組み込まれた、2 行に分割された右から左のテキストは、次のように書かれます。

```
THERE IS senil owt neewteb tilps si taht txet lanoitceridib deddebme IN THIS SENTENCE.
```

両方のテキストとも、組み込みテキストが分割されていても、読みやすさを保っています。

関連概念:

12 ページの『両方向性と文字成形』
国際化されたプログラムが、テキストの両方向性や字形を処理する必要が生じることがあります。

データ・ストリーム

このセクションでは、両方向テキスト環境用のデータ・ストリームについて説明します。

両方向テキスト環境では、以下のデータ・ストリームを使用します。

データ・ストリーム	説明

ビジュアル・データ・ストリーム	システムは、文字を、スクリーンで表示される順序で編成します。
	
	ビジュアル・データ・ストリームが左から右に表示されると、データ・ストリームの先頭文字は、ビューポート (スクリーン、ウィンドウ、行、フィールド、等々) の左側に来ます。 同じデータ・ストリームが右から左のビューポートに表示される場合、データ・ストリームの最初の文字は右に来ます。
	
	ビジュアル・データ・ストリームに、書く方向が反対の言語が組み込まれると、各テキストの順序は、ビューポートの方向が反転したとき、保たれます。例えば、(小文字テキストが両方向テキストを表す) キー・ストローク順序が次のようになっている場合、
	
	`THERE IS bidirectional text IN THIS SENTENCE.`
	
	ビジュアル・データ・ストリームは、次のようになります。
	
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	
	このビジュアル・データ・ストリームの左から右ビューポートでの表示は、次のように、左揃えになります。
	
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	`-------> <----------------- ---------------->`
	
	矢印は、読み取り方向を示します。
	
	ビューポート方向を右から左に変えると、次のように、ビジュアル・データ・ストリームは反転し、右揃えとなって、読みにくくなります。
	
	`.ECNETNES SIHT NI bidirectional text SI EREHT`
	`<---------------- -----------------> <-------`
	
	したがって、英語テキストがアラビア語またはヘブライ語のテキスト内に組み込まれると、両テキストとも、正しい読み取り順序になるのは、左から右のビューポートの場合に限られます。 英語にアラビア語またはヘブライ語が組み込まれた場合にも、同じことが当てはまります。 ビューポート方向を逆にすると、両テキストとも読みにくくなります。

データ・ストリーム	説明

論理データ・ストリーム	システムは、文字を読みやすい順序で編成します。 両方向表示管理機能は、テキスト・ストリングを読みやすい順序に配置します。
	
	論理データ・ストリームが左から右のビューポートに表示されると、データ・ストリームの最初の文字は左側に表示されます。 同じデータ・ストリームが右から左のビューポートに表示されると、データ・ストリームの最初の文字は右側に表示されますが、読みやすい表示順序であることは依然変わりません。
	
	論理データ・ストリームに、書く方向が反対の言語が組み込まれても、各テキストの方向は、両方向表示管理機能によって保たれます。 例えば、キー・ストローク順序が次のようになっている場合は、
	`THERE IS bidirectional text IN THIS SENTENCE.`
	
	論理データ・ストリームは同じです。 例えば、次のようになります。
	`THERE IS bidirectional text IN THIS SENTENCE.`
	
	この論理データ・ストリームの左から右のビューポートでの表示 (左揃え) は、以下のようになります。
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	`-------> <----------------- ----------------->`
	
	論理データ・ストリームの右から左のビューポートでの表示 (右揃え) は、次のようになります。
	`IN THIS SENTENCE. txet lanoitceridib THERE IS`
	`----------------> <----------------- ------->`
	
	どちらのビューポート方向でも、読みやすい論理データ・ストリームになります。

カーソル移動

このセクションでは、スクリーンが両方向テキストを含む場合の、スクリーン上のカーソル移動について説明します。

両方向テキストを含むスクリーンでのカーソル移動は、以下のようになります。

カーソル移動	説明

ビジュアル	カーソルは、現在位置から、左または右の次の文字、または上または下の次の行に移動します。 例えば、カーソルが、次のように混合文の最初の左から右の部分の終わりにあるとします。
	`THERE IS_txet lanoitceridib IN THIS SENTENCE.`
	
	ここで、カーソルをビジュアルに右に移すと、カーソルは次のように、1 文字右に移動します。
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	
	カーソルは、テキストの内容に関係なく移動します。

カーソル移動	説明

論理	カーソルは、データ・ストリーム内の現在位置から次の文字か前の文字に移動します。 その文字は、同じ行の別の位置か、スクリーン上の別の行上で、カーソル位置に隣接していることもあります。 カーソルを論理的に移動するには、データ・ストリームをスキャンして、次の論理文字を検出する必要があります。 例えば、カーソルが、次のように混合文の最初の左から右の部分の終わりにあるとします。
	`THERE IS_txet lanoitceridib IN THIS SENTENCE.`
	ここで、カーソルを論理的に次の文字に移動すると、データ・ストリームのスキャンが行われ、次の論理文字が検出されます。 次のように、カーソルは文の次の論理部分に移動します。
	`THERE IS txet lanoitceridib_IN THIS SENTENCE.`
	カーソルは、内容に従って移動します。

字形

字形は、文字の形状が、テキスト行におけるその位置によって決まるときに発生します。 アラビア語のように、言語によっては、文字の形状がストリング内のその位置や、周囲の文字によって変わるものもあります。

以下の特性が、アラビア語スクリプトの字形を決定します。

- 書かれる言語に、大文字に相当する文字はない。
- 文字の形状は、その文字のストリング内での位置と、周囲の文字によって変わる。
- 書かれる言語は筆記体である。 ワードのほとんどの文字が、英語の筆記のように接しています。
- 文字を結合して、スペースなしの文字にすることができる。 さらに、文字の上か下に母音または発音区別符号のマークを書くことができます。
- 文字の長さを変えて、2 つのコード化形状で出力することができる。

文字成形の方法**:**

字形を、その他のシステム・コンポーネントとは関係なくインプリメントします。 しかし、字形には、その他のシステム・コンポーネントがユーティリティーとしてアクセスすることができます。

システムは、次の方法で字形を使用することができます。

- ユーザーがデータをコンピューターに入力する際、システムが字形を使用して、文字を形成します。 システムは、これらの文字をその形成済みの形式で保管します。

  この方法では、これらの文字を表示するたびに字形を使用する必要はありません。 この方法の意図は、メニューやヘルプなどの静的データを対象にしています。 この方法では、文字を正しくソートしたり、検索したり、あるいは索引付けするためのプリプロセスが必要です。

  文字は、正しく表示するために、処理後に再形成を必要とする場合があります。
- ユーザーがデータをコンピューターに入力する際、システムはその文字を未形成の形式で保管します。

  この方法を用いると、文字をソートしたり、検索したり、あるいは索引付けすることができます。 ただし、システムは文字を表示するたびに字形を使用しなければなりません。

基本形状は、字形によって生成されなかった分離形状です。 文字ストリングの編集、検索またはその他のテキスト操作の際は、基本形状を使用します。 形成を使用するのは、テキストの表示または印刷のときに限られます。 文字がその形成された形式で保管されていると、システムは、それを形成解除しなければ、

ソート、照合、検索、または索引付けすることができません。ストリング内の位置によって決められた形状ではない字形は、異なるコーディング環境との通信だけでなく、特定の文字処理アプリケーションにとって必要です。

コンテキスト字形:

一般に、コンテキスト字形は、必要な文字の形状を、ワード内での位置と周囲の文字によって、与えられたフォント内で選択することです。

以下に、あり得る形状を示します。

字形	説明
分離	前後の文字に接していない文字
最終	前の文字に接しているが、後の文字には接していない文字
最初	後の文字に接しているが、前の文字には接していない文字
中間	前後両方の文字に接している文字

文字は、以下の特性のいずれかを持つこともできます。

- 前の文字に接する
- 後の文字に接する
- 周囲の文字が接することができる

頭字語、部品番号およびグラフィック文字は、コンテキスト字形を必要としません。 これらの文字を正しく入力するには、コンテキスト字形をオフにし、特定のキーボード・インターフェースを用いて、希望する形状を正確に選択してください。 これらの文字には、後で表示できるように、フィールド、行、または制御文字のタグを付けます。

サポートされる言語およびロケール

フル・クライアント・モードでサポートされている多くのロケールに加えて、AIX では多くの新しい Unicode ロケールに対して「サーバー・サイド・サポート」を提供しています。

サーバー・サイド・ロケールのサポートでは、以下の機能が提供されます。

- Unicode ストリングの保管および操作
- 変換機能 (iconv)
- 数値、通貨、日付、時刻、およびメッセージの形式設定および構文解析関数
- メッセージをさまざまな言語に翻訳し、ユーザーのロケールに基づいてプログラムで使用できるようにするメッセージ機能
- Unicode 準拠の照合、正規化、およびテキスト境界アルゴリズム
- 同一アプリケーションにおける複数のロケールの同時使用
- 任意のロケールですべての Unicode 文字を使用できる、コード・セットから独立したロケール

以下の機能はサーバー・サイド・ロケール・サポートでは提供されていません。これらの機能はクライアントで提供されています。

- グラフィカル・ユーザー・インターフェース
- ユーザー入力 (キーボード・マッピング、入力メソッドなど)
- 最終出力 (表示、フォント、字形、印刷など)

グローバリゼーション・サポートでは、ロケール定義を含むソース・ファイルは、さまざまな標準が原因で異なったり不整合になったりする場合があります。2 つの主要なロケール定義 (IBM および Unicode の Common Locale Data Repository (CLDR) 定義) は、ロケール・オブジェクトの作成に使用されます。

表 *15. AIX* での *Unicode* エンコード済み言語およびロケール

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト **LFT** キーボード **ID**	**CLDR locale?**	**AIX** リリースでの導入
アフリカーンス語	南アフリカ	UTF-8	af_ZA.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
アルバニア語	アルバニア	UTF-8	SQ_AL	SQ_AL.UTF-8	いいえ	はい	452	いいえ	5.2 以前
アルバニア語	アルバニア	UTF-8	sq_AL.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アムハラ語	エチオピア	UTF-8	am_ET.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
アラビア語	アルジェリア	UTF-8	AR_DZ	AR_DZ.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	アルジェリア	UTF-8	ar_DZ.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	バーレーン	UTF-8	AR_BH	AR_BH.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	バーレーン	UTF-8	ar_BH.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	デフォルト	UTF-8	AR_AA	AR_AA.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	エジプト	UTF-8	AR_EG	AR_EG.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	エジプト	UTF-8	ar_EG.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	イラク共和国	UTF-8	ar_IQ.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
アラビア語	ヨルダン	UTF-8	AR_JO	AR_JO.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	ヨルダン	UTF-8	ar_JO.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	クウェート	UTF-8	AR_KW	AR_KW.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	クウェート	UTF-8	ar_KW.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	レバノン	UTF-8	AR_LB	AR_LB.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	レバノン	UTF-8	ar_LB.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	リビア	UTF-8	ar_LY.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
アラビア語	モーリタニア	UTF-8	ar_MR.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
アラビア語	モロッコ	UTF-8	AR_MA	AR_MA.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	モロッコ	UTF-8	ar_MA.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	オマーン	UTF-8	AR_OM	AR_OM.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	オマーン	UTF-8	ar_OM.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	カタール	UTF-8	AR_QA	AR_QA.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	カタール	UTF-8	ar_QA.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	サウジアラビア	UTF-8	AR_SA	AR_SA.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	サウジアラビア	UTF-8	ar_SA.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	シリア	UTF-8	AR_SY	AR_SY.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	シリア	UTF-8	ar_SY.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	チュニジア	UTF-8	AR_TN	AR_TN.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	チュニジア	UTF-8	ar_TN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	アラブ首長国連邦	UTF-8	AR_AE	AR_AE.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	アラブ首長国連邦	UTF-8	ar_AE.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アラビア語	イエメン	UTF-8	AR_YE	AR_YE.UTF-8	いいえ	はい	253	いいえ	5.2 以前
アラビア語	イエメン	UTF-8	ar_YE.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
アルメニア語	アルメニア共和国	UTF-8	hy_AM.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
アッサム語	インド	UTF-8	AS_IN	AS_IN.UTF-8	いいえ	はい	485	はい	5.3
アッサム語	インド	UTF-8	as_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
アゼルバイジャン語 (ローマ字)	アゼルバイジャン	UTF-8	AZ_AZ	AZ_AZ.UTF-8	いいえ	はい	490	はい	6.1
アゼルバイジャン語 (ローマ字)	アゼルバイジャン	UTF-8	az_Latn_AZ.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
バスク語	スペイン	UTF-8	eu_ES.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ベラルーシ語	ベラルーシ共和国	UTF-8	BE_BY	BE_BY.UTF-8	いいえ	はい	463	いいえ	5.2 以前
ベラルーシ語	ベラルーシ共和国	UTF-8	be_BY.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ベンガル語	バングラデシュ	UTF-8	bn_BD.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ベンガル語	インド	UTF-8	BN_IN	BN_IN.UTF-8	いいえ	はい	480	はい	5.3
ベンガル語	インド	UTF-8	bn_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ボスニア語	ボスニア	UTF-8	bs_Latn_BA.UTF-8		いいえ	いいえ	103P	はい	7.2.1.0
ブルガリア語	ブルガリア	UTF-8	BG_BG	BG_BG.UTF-8	いいえ	はい	442	いいえ	5.2 以前
ブルガリア語	ブルガリア	UTF-8	bg_BG.UTF-8		いいえ	いいえ	103P	はい	7.1.20
ビルマ語	ミャンマー	UTF-8	my_MM.UTF-8		いいえ	いいえ	103P	はい	7.2.1.0
カタロニア語	スペイン	UTF-8	CA_ES	CA_ES.UTF-8	はい	はい	173	いいえ	5.2 以前
カタロニア語	スペイン	UTF-8	ca_ES.UTF-8		はい	いいえ	103P	はい	7.1.2.0
中国語 (簡体字)	中華人民共和国	UTF-8	ZH_CN	ZH_CN.UTF-8	はい	はい	103P	いいえ	5.2 以前
中国語 (簡体字)	中華人民共和国	UTF-8	zh_Hans_CN.UTF-8		はい	いいえ	103P	はい	7.1.2.0
中国語 (簡体字)	香港特別行政区	UTF-8	ZH_HK	ZH_HK.UTF-8	はい	はい	467	いいえ	5.2 以前
中国語 (簡体字)	シンガポール	UTF-8	ZH_SG	ZH_SG.UTF-8	はい	はい	103P	いいえ	5.2 以前
中国語 (簡体字)	シンガポール	UTF-8	zh_Hans_SG.UTF-8		はい	いいえ	103P	はい	7.1.2.0

表 15. AIX での Unicode エンコード済み言語およびロケール (続き)

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト LFT キーボード ID	CLDR locale?	AIX リリースでの導入
中国語 (繁体字)	香港特別行政区	UTF-8	zh_Hant_HK.UTF-8		はい	いいえ	103P	はい	7.1.2.0
中国語 (繁体字)	マカオ	UTF-8	zh_Hant_MO.UTF-8		はい	いいえ	103P	はい	7.2.1.0
中国語 (繁体字)	台湾	UTF-8	ZH_TW	ZH_TW.UTF-8	はい	はい	467	いいえ	5.2 以前
中国語 (繁体字)	台湾	UTF-8	zh_Hant_TW.UTF-8		はい	いいえ	103P	はい	7.1.2.0
クロアチア語	クロアチア	UTF-8	HR_HR	HR_HR.UTF-8	いいえ	はい	234	いいえ	5.2 以前
クロアチア語	クロアチア	UTF-8	hr_HR.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
チェコ語	チェコ	UTF-8	CS_CZ	CS_CZ.UTF-8	はい	はい	234	いいえ	5.2 以前
チェコ語	チェコ	UTF-8	cs_CZ.UTF-8		はい	いいえ	103P	はい	7.1.2.0
デンマーク語	デンマーク	UTF-8	DA_DK	DA_DK.UTF-8	いいえ	はい	159	いいえ	5.2 以前
デンマーク語	デンマーク	UTF-8	da_DK.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
オランダ語	ベルギー	UTF-8	NL_BE	NL_BE.UTF-8	いいえ	はい	120	いいえ	5.2 以前
オランダ語	ベルギー	UTF-8	nl_BE.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
オランダ語	オランダ	UTF-8	NL_NL	NL_NL.UTF-8	いいえ	はい	143	いいえ	5.2 以前
オランダ語	オランダ	UTF-8	nl_NL.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
英語	オーストラリア	UTF-8	EN_AU	EN_AU.UTF-8	はい	はい	103P	いいえ	5.2 以前
英語	オーストラリア	UTF-8	en_AU.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	ベルギー	UTF-8	EN_BE	EN_BE.UTF-8	はい	はい	120	いいえ	5.2 以前
英語	ベルギー	UTF-8	en_BE.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	カナダ	UTF-8	EN_CA	EN_CA.UTF-8	はい	はい	445	いいえ	5.2 以前
英語	カナダ	UTF-8	en_CA.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	カメルーン	UTF-8	en_CM.UTF-8		はい	いいえ	103P	はい	7.1.4.0
英語	ガーナ	UTF-8	en_GH.UTF-8		はい	いいえ	103P	はい	7.1.4.0
英語	香港特別行政区	UTF-8	EN_HK	EN_HK.UTF-8	はい	はい	168	いいえ	5.2 以前
英語	香港特別行政区	UTF-8	en_HK.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	ケニア共和国	UTF-8	en_KE.UTF-8		はい	いいえ	103P	はい	7.1.4.0
英語	インド	UTF-8	EN_IN	EN_IN.UTF-8	はい	はい	468	いいえ	5.2 以前
英語	インド	UTF-8	en_IN.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	アイルランド	UTF-8	EN_IE	EN_IE.UTF-8	はい	はい	168	いいえ	5.2 以前
英語	アイルランド	UTF-8	en_IE.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	モーリシャス	UTF-8	en_MU.UTF-8		はい	いいえ	103P	はい	7.1.4.0
英語	ニュージーランド	UTF-8	EN_NZ	EN_NZ.UTF-8	はい	はい	103P	いいえ	5.2 以前
英語	ニュージーランド	UTF-8	en_NZ.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	ナイジェリア連邦共和国	UTF-8	en_NG.UTF-8		はい	いいえ	103P	はい	7.1.4.0
英語	フィリピン	UTF-8	EN_PH	EN_PH.UTF-8	はい	はい	103P	いいえ	5.2 以前
英語	フィリピン	UTF-8	en_PH.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	シンガポール	UTF-8	EN_SG	EN_SG.UTF-8	はい	はい	168	いいえ	5.2 以前
英語	シンガポール	UTF-8	en_SG.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	南アフリカ	UTF-8	EN_ZA	EN_ZA.UTF-8	はい	はい	103P	いいえ	5.2 以前
英語	南アフリカ	UTF-8	en_ZA.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	タンザニア連合共和国	UTF-8	en_TZ.UTF-8		はい	いいえ	103P	はい	7.1.4.0
英語	グレートブリテンおよび北部アイルランド連合王国 (英国)	UTF-8	EN_GB	EN_GB.UTF-8	はい	はい	166	いいえ	5.2 以前
英語	グレートブリテンおよび北部アイルランド連合王国 (英国)	UTF-8	en_GB.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	アメリカ合衆国	UTF-8	EN_US	EN_US.UTF-8	はい	はい	103P	いいえ	5.2 以前
英語	アメリカ合衆国	UTF-8	en_US.UTF-8		はい	いいえ	103P	はい	7.1.2.0
英語	ザンビア	UTF-8	en_ZM.UTF-8		はい	いいえ	103P	はい	7.1.4.0
エストニア語	エストニア	UTF-8	ET_EE	ET_EE.UTF-8	いいえ	はい	454	いいえ	5.3
エストニア語	エストニア	UTF-8	et_EE.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
フィリピノ語 (ローマ字)	フィリピン	UTF-8	fil_PH.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
フィンランド語	フィンランド	UTF-8	FI_FI	FI_FI.UTF-8	いいえ	はい	153	いいえ	5.2 以前
フィンランド語	フィンランド	UTF-8	fi_FI.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
フランス語	アルジェリア	UTF-8	fr_DZ.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	ベルギー	UTF-8	FR_BE	FR_BE.UTF-8	はい	はい	120	いいえ	5.2 以前
フランス語	ベルギー	UTF-8	fr_BE.UTF-8		はい	いいえ	103P	はい	7.1.2.0
フランス語	カナダ	UTF-8	FR_CA	FR_CA.UTF-8	はい	はい	58	いいえ	5.2 以前
フランス語	カナダ	UTF-8	fr_CA.UTF-8		はい	いいえ	103P	はい	7.1.2.0
フランス語	カメルーン	UTF-8	fr_CM.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	コンゴ民主共和国	UTF-8	fr_CD.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	フランス	UTF-8	FR_FR	FR_FR.UTF-8	はい	はい	120	いいえ	5.2 以前
フランス語	フランス	UTF-8	fr_FR.UTF-8		はい	いいえ	103P	はい	7.1.2.0
フランス語	ルクセンブルグ	UTF-8	FR_LU	FR_LU.UTF-8	はい	はい	120	いいえ	5.2 以前

表 15. AIX での Unicode エンコード済み言語およびロケール (続き)

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト LFT キーボード ID	CLDR locale?	AIX リリースでの導入
フランス語	ルクセンブルグ	UTF-8	fr_LU.UTF-8		はい	いいえ	103P	はい	7.1.2.0
フランス語	コートジボアール	UTF-8	fr_CI.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	モーリタニア	UTF-8	fr_MR.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	モーリシャス	UTF-8	fr_MU.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	モロッコ	UTF-8	fr_MA.UTF-8		はい	いいえ	103P	はい	7.1.4.0
フランス語	セネガル	UTF-8	fr_SN.UTF-8		はい	いいえ	103P	はい	7.1.3.0
フランス語	スイス	UTF-8	FR_CH	FR_CH.UTF-8	はい	はい	150F	いいえ	5.2 以前
フランス語	スイス	UTF-8	fr_CH.UTF-8		はい	いいえ	103P	はい	7.1.2.0
フランス語	チュニジア	UTF-8	fr_TN.UTF-8		はい	いいえ	103P	はい	7.1.4.0
ガリシア語	スペイン	UTF-8	gl_ES.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ガンダ語	ウガンダ共和国	UTF-8	lg_UG.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
グルジア語	グルジア共和国	UTF-8	ka_GE.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ドイツ語	オーストリア	UTF-8	DE_AT	DE_AT.UTF-8	はい	はい	129	いいえ	5.2 以前
ドイツ語	オーストリア	UTF-8	de_AT.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ドイツ語	ドイツ	UTF-8	DE_DE	DE_DE.UTF-8	はい	はい	129	いいえ	5.2 以前
ドイツ語	ドイツ	UTF-8	de_DE.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ドイツ語	ルクセンブルグ	UTF-8	DE_LU	DE_LU.UTF-8	はい	はい	150G	いいえ	5.2 以前
ドイツ語	ルクセンブルグ	UTF-8	de_LU.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ドイツ語	スイス	UTF-8	DE_CH	DE_CH.UTF-8	はい	はい	150G	いいえ	5.2 以前
ドイツ語	スイス	UTF-8	de_CH.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ギリシャ語	ギリシャ	UTF-8	EL_GR	EL_GR.UTF-8	いいえ	はい	319	いいえ	5.2 以前
ギリシャ語	ギリシャ	UTF-8	el_GR.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
グジャラート語	インド	UTF-8	GU_IN	GU_IN.UTF-8	いいえ	はい	477	いいえ	5.3
グジャラート語	インド	UTF-8	gu_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ハウサ語	ナイジェリア連邦共和国	UTF-8	ha_Latn_NG.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ヘブライ語	イスラエル	UTF-8	HE_IL	HE_IL.UTF-8	いいえ	はい	115	いいえ	5.2 以前
ヘブライ語	イスラエル	UTF-8	he_IL.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ヒンディ語	インド	UTF-8	HI_IN	HI_IN.UTF-8	いいえ	はい	468	いいえ	5.2 以前
ヒンディ語	インド	UTF-8	hi_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ハンガリー語	ハンガリー	UTF-8	HU_HU	HU_HU.UTF-8	はい	はい	208	いいえ	5.2 以前
ハンガリー語	ハンガリー	UTF-8	hu_HU.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ラオ語	ラオス	UTF-8	lo_LA.UTF-8	いいえ	いいえ		103P		7.2.1.0
アイスランド語	アイスランド	UTF-8	IS_IS	IS_IS.UTF-8	いいえ	はい	197	いいえ	5.2 以前
アイスランド語	アイスランド	UTF-8	is_IS.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
イグボ語	ナイジェリア連邦共和国	UTF-8	ig_NG.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
インドネシア語	インドネシア	UTF-8	ID_ID	ID_ID.UTF-8	いいえ	はい	103P	いいえ	5.2 以前
インドネシア語	インドネシア	UTF-8	id_ID.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
イタリア語	イタリア	UTF-8	IT_IT	IT_IT.UTF-8	はい	はい	142	いいえ	5.2 以前
イタリア語	イタリア	UTF-8	it_IT.UTF-8		はい	いいえ	103P	はい	7.1.2.0
イタリア語	スイス	UTF-8	IT_CH	IT_CH.UTF-8	はい	はい	150G	いいえ	5.2 以前
イタリア語	スイス	UTF-8	it_CH.UTF-8		はい	いいえ	103P	はい	7.1.2.0
日本語	日本	UTF-8	JA_JP	JA_JP.UTF-8	はい	はい	194	いいえ	5.2 以前
日本語	日本	UTF-8	ja_JP.UTF-8		はい	いいえ	103P	はい	7.1.2.0
カンナダ語	インド	UTF-8	KN_IN	KN_IN.UTF-8	いいえ	はい	483	はい	5.3
カンナダ語	インド	UTF-8	kn_IN.UTF-8	8	いいえ	いいえ	103P	はい	7.1.4.0
カザフ語	カザフスタン	UTF-8	KK_KZ	KK_KZ.UTF-8	いいえ	はい	476	いいえ	5.3
カザフ語	カザフスタン	UTF-8	kk_Cyrl_KZ.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
キニヤルワンダ語	ルワンダ共和国	UTF-8	rw_RW.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
クメール語	カンボジア	UTF-8	km_KH.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
コンカニー語	インド	UTF-8	kok_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
韓国語	韓国	UTF-8	KO_KR	KO_KR.UTF-8	はい	はい	413	いいえ	5.2 以前
韓国語	韓国	UTF-8	ko_KR.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ラトビア語	ラトビア	UTF-8	LV_LV	LV_LV.UTF-8	いいえ	はい	455	いいえ	5.2 以前
ラトビア語	ラトビア	UTF-8	lv_LV.UTF-8		いいえ	はい	103P	はい	7.1.2.0
リトアニア語	リトアニア共和国	UTF-8	LT_LT	LT_LT.UTF-8	いいえ	はい	456	いいえ	5.2 以前
リトアニア語	リトアニア共和国	UTF-8	lt_LT.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
マケドニア語	マケドニア	UTF-8	MK_MK	MK_MK.UTF-8	いいえ	はい	449	いいえ	5.2 以前
マケドニア語	マケドニア	UTF-8	mk_MK.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
マレー語	マレーシア	UTF-8	MS_MY	MS_MY.UTF-8	いいえ	はい	103P	いいえ	5.2 以前
マレー語	マレーシア	UTF-8	ms_MY.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
マラヤーラム文字	インド	UTF-8	ML_IN	ML_IN.UTF-8	いいえ	はい	479	いいえ	5.3
マラヤーラム文字	インド	UTF-8	ml_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
マルタ語	マルタ	UTF-8	MT_MT	MT_MT.UTF-8	いいえ	はい	491	いいえ	6.1

表 15. AIX での Unicode エンコード済み言語およびロケール (続き)

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト LFT キーボード ID	CLDR locale?	AIX リリースでの導入
マルタ語	マルタ	UTF-8	mt_MT.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
マラーティー語	インド	UTF-8	MR_IN	MR_IN.UTF-8	いいえ	はい	468	いいえ	5.3
マラーティー語	インド	UTF-8	mr_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
モンゴル語 (キリル文字)	モンゴル	UTF-8	mn_Cyrl_MN.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ネパール語	インド	UTF-8	ne_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ネパール語	ネパール王国	UTF-8	ne_NP.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ノルウェー語ブークモール	ノルウェー	UTF-8	nb_NO.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ノルウェー語ニーノシュク	ノルウェー	UTF-8	nn_NO.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
オリヤー語	インド	UTF-8	OR_IN	OR_IN.UTF-8	いいえ	はい	482	はい	5.3
オリヤー語	インド	UTF-8	or_IN.UTF-8		いいえ	いいえ	103P	はい	7.4.1.0
オロモ語	エチオピア	UTF-8	om_ET.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ポーランド語	ポーランド	UTF-8	PL_PL	PL_PL.UTF-8	はい	はい	214	いいえ	5.2 以前
ポーランド語	ポーランド	UTF-8	pl_PL.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ポルトガル語	アンゴラ	UTF-8	pt_AO.UTF-8		はい	いいえ	103P	はい	7.1.4.0
ポルトガル語	ブラジル	UTF-8	PT_BR	PT_BR.UTF-8	はい	はい	275	いいえ	5.2 以前
ポルトガル語	ブラジル	UTF-8	pt_BR.UTF-8		はい	いいえ	103P	はい	7.1.2.0
ポルトガル語	マカオ	UTF-8	pt_MO.UTF-8		いいえ	いいえ	103P	はい	7.2.1.0
ポルトガル語	モザンビーク	UTF-8	pt_MZ.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ポルトガル語	ポルトガル語	UTF-8	PT_PT	PT_PT.UTF-8	いいえ	はい	163	いいえ	5.2 以前
ポルトガル語	ポルトガル語	UTF-8	pt_PT.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
パンジャブ語	インド	UTF-8	PA_IN	PA_IN.UTF-8	いいえ	はい	484	はい	5.3
パンジャブ語	インド	UTF-8	pa_Guru_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ルーマニア語	ルーマニア	UTF-8	RO_RO	RO_RO.UTF-8	いいえ	はい	446	いいえ	5.2 以前
ルーマニア語	ルーマニア	UTF-8	ro_RO.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ロシア語	ロシア	UTF-8	RU_RU	RU_RU.UTF-8	はい	はい	441	いいえ	5.2 以前
ロシア語	ロシア	UTF-8	ru_RU.UTF-8		はい	いいえ	103P	はい	7.1.2.0
セルビア語 (ローマ字)	モンテネグロ	UTF-8	sr_Latn_ME.UTF-8		いいえ	いいえ	103P	はい	7.2.1.0
セルビア語 (キリル文字)	セルビア	UTF-8	sr_Cyrl_RS.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
セルビア語 (キリル文字)	セルビア	UTF-8	SR_SP	SR_SP.UTF-8	いいえ	いいえ	450	いいえ	5.2 以前
セルビア語 (ローマ字)	セルビア	UTF-8	sr_Latn_RS.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
セルビア語 (ローマ字)	セルビア	UTF-8	SH_SP	SH_SP.UTF-8	いいえ	いいえ	234	いいえ	5.2 以前
シンハラ語	スリランカ民主社会主義共和国	UTF-8	si_LK.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
スロバキア語	スロバキア	UTF-8	SK_SK	SK_SK.UTF-8	はい	はい	245	いいえ	5.2 以前
スロバキア語	スロバキア	UTF-8	sk_SK.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スロベニア語	スロベニア	UTF-8	SL_SI	SL_SI.UTF-8	いいえ	はい	234	いいえ	5.2 以前
スロベニア語	スロベニア	UTF-8	sl_SI.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
スペイン語	アルゼンチン	UTF-8	ES_AR	ES_AR.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	アルゼンチン	UTF-8	es_AR.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ボリビア	UTF-8	ES_BO	ES_BO.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ボリビア	UTF-8	es_BO.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	チリ	UTF-8	ES_CL	ES_CL.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	チリ	UTF-8	es_CL.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	コロンビア	UTF-8	ES_CO	ES_CO.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	コロンビア	UTF-8	es_CO.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	コスタリカ	UTF-8	ES_CR	ES_CR.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	コスタリカ	UTF-8	es_CR.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ドミニカ共和国	UTF-8	ES_DO	ES_DO.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ドミニカ共和国	UTF-8	es_DO.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	エクアドル	UTF-8	ES_EC	ES_EC.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	エクアドル	UTF-8	es_EC.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	エルサルバドル共和国	UTF-8	ES_SV	ES_SV.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	エルサルバドル共和国	UTF-8	es_SV.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	グアテマラ	UTF-8	ES_GT	ES_GT.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	グアテマラ	UTF-8	es_GT.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ホンジュラス	UTF-8	ES_HN	ES_HN.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ホンジュラス	UTF-8	es_HN.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	メキシコ	UTF-8	ES_MX	ES_MX.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	メキシコ	UTF-8	es_MX.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ニカラグア	UTF-8	ES_NI	ES_NI.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ニカラグア	UTF-8	es_NI.UTF-8		はい	いいえ	103P	はい	7.1.2.0

表 *15. AIX* での *Unicode* エンコード済み言語およびロケール *(*続き*)*

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト **LFT** キーボード **ID**	**CLDR locale?**	**AIX** リリースでの導入
スペイン語	パナマ	UTF-8	ES_PA	ES_PA.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	パナマ	UTF-8	es_PA.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	パラグアイ	UTF-8	ES_PY	ES_PY.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	パラグアイ	UTF-8	es_PY.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ペルー	UTF-8	ES_PE	ES_PE.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ペルー	UTF-8	es_PE.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	プエルトリコ	UTF-8	ES_PR	ES_PR.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	プエルトリコ	UTF-8	es_PR.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	スペイン	UTF-8	ES_ES	ES_ES.UTF-8	はい	はい	173	いいえ	5.2 以前
スペイン語	スペイン	UTF-8	es_ES.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	アメリカ合衆国	UTF-8	ES_US	ES_US.UTF-8	はい	はい	103P	いいえ	5.2 以前
スペイン語	アメリカ合衆国	UTF-8	es_US.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ウルグアイ	UTF-8	ES_UY	ES_UY.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ウルグアイ	UTF-8	es_UY.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スペイン語	ベネズエラ	UTF-8	ES_VE	ES_VE.UTF-8	はい	はい	171	いいえ	5.2 以前
スペイン語	ベネズエラ	UTF-8	es_VE.UTF-8		はい	いいえ	103P	はい	7.1.2.0
スワヒリ語	ケニア共和国	UTF-8	sw_KE.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
スワヒリ語	タンザニア連合共和国	UTF-8	sw_TZ.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
スウェーデン語	スウェーデン	UTF-8	SV_SE	SV_SE.UTF-8	いいえ	はい	153	いいえ	5.2 以前
スウェーデン語	スウェーデン	UTF-8	sv_SE.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
タミール語	インド	UTF-8	TA_IN	TA_IN.UTF-8	いいえ	はい	474	いいえ	5.3
タミール語	インド	UTF-8	ta_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
テルグ語	インド	UTF-8	TE_IN	TE_IN.UTF-8	いいえ	はい	473	いいえ	5.3
テルグ語	インド	UTF-8	te_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
タイ語	タイ	UTF-8	TH_TH	TH_TH.UTF-8	いいえ	はい	191	いいえ	5.2 以前
タイ語	タイ	UTF-8	th_TH.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
トルコ語	トルコ	UTF-8	TR_TR	TR_TR.UTF-8	いいえ	はい	179	いいえ	5.2 以前
トルコ語	トルコ	UTF-8	tr_TR.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ウクライナ語	ウクライナ	UTF-8	UK_UA	UK_UA.UTF-8	いいえ	はい	465	いいえ	5.2 以前
ウクライナ語	ウクライナ	UTF-8	uk_UA.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ウルドゥー語	インド	UTF-8	UR_IN	UR_IN.UTF-8	いいえ	はい	492	はい	6.1
ウルドゥー語	インド	UTF-8	ur_IN.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ウルドゥー語	パキスタン	UTF-8	UR_PK	UR_PK.UTF-8	いいえ	はい	492	はい	6.1
ウルドゥー語	パキスタン	UTF-8	ur_PK.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ウズベク語	ウズベキスタン	UTF-8	uz_Cyrl_UZ.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ウズベク語	ウズベキスタン	UTF-8	uz_Latn_UZ.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ベトナム語	ベトナム	UTF-8	VI_VN	VI_VN.UTF-8	いいえ	はい	461	いいえ	5.2 以前
ベトナム語	ベトナム	UTF-8	vi_VN.UTF-8		いいえ	いいえ	103P	はい	7.1.2.0
ウェールズ語	グレートブリテンおよび北部アイルランド連合王国 (英国)	UTF-8	CY_GB	CY_GB.UTF-8	いいえ	はい	166W	はい	6.1
ウェールズ語	グレートブリテンおよび北部アイルランド連合王国 (英国)	UTF-8	cy_GB.UTF-8		いいえ	いいえ	103P	はい	7.1.4.0
ヨルバ語 (ローマ字)	ナイジェリア連邦共和国	UTF-8	yo_NG.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0
ズールー語	南アフリカ	UTF-8	zu_ZA.UTF-8		いいえ	いいえ	103P	はい	7.1.1.0

表 *16. AIX* での非 *Unicode* エンコード済み言語およびロケール

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト **LFT** キーボード **ID**	**CLDR locale?**	**AIX** リリースでの導入
アルバニア語	アルバニア	ISO8859-15	sq_AL.8859-15		いいえ	はい	452	いいえ	5.2 以前
アルバニア語	アルバニア	ISO8859-1	sq_AL	sq_AL.ISO8859-1	いいえ	はい	452	いいえ	5.2 以前
アラビア語	アルジェリア	ISO8859-6	ar_DZ	ar_DZ.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	バーレーン	ISO8859-6	ar_BH	ar_BH.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	デフォルト	ISO8859-6	ar_AA	ar_AA.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	デフォルト	IBM-1046	Ar_AA	Ar_AA.IBM-1046	いいえ	はい	253	いいえ	5.2 以前
アラビア語	エジプト	ISO8859-6	ar_EG	ar_EG.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	ヨルダン	ISO8859-6	ar_JO	ar_JO.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	クウェート	ISO8859-6	ar_KW	ar_KW.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	レバノン	ISO8859-6	ar_LB	ar_LB.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	モロッコ	ISO8859-6	ar_MA	ar_MA.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	オマーン	ISO8859-6	ar_OM	ar_OM.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	カタール	ISO8859-6	ar_QA	ar_QA.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	サウジアラビア	ISO8859-6	ar_SA	ar_SA.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	シリア	ISO8859-6	ar_SY	ar_SY.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前

表 16. *AIX* での非 *Unicode* エンコード済み言語およびロケール (続き)

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト **LFT** キーボード **ID**	**CLDR locale?**	**AIX** リリースでの導入
アラビア語	チュニジア	ISO8859-6	ar_TN	ar_TN.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	アラブ首長国連邦	ISO8859-6	ar_AE	ar_AE.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
アラビア語	イエメン	ISO8859-6	ar_YE	ar_YE.ISO8859-6	いいえ	はい	253	いいえ	5.2 以前
ベラルーシ語	ベラルーシ共和国	ISO8859-5	be_BY	be_BY.ISO8859-5	いいえ	はい	463	いいえ	5.2 以前
ブルガリア語	ブルガリア	ISO8859-5	bg_BG	bg_BG.ISO8859-5	いいえ	はい	442	いいえ	5.2 以前
カタロニア語	スペイン	ISO8859-15	ca_ES.8859-15		はい	はい	173	いいえ	5.2 以前
カタロニア語	スペイン	ISO8859-1	ca_ES	ca_ES.ISO8859-1	はい	はい	173	いいえ	5.2 以前
カタロニア語	スペイン	IBM-1252	ca_ES.IBM-1252		はい	はい	173	いいえ	5.2 以前
中国語 (簡体字)	中華人民共和国	IBM-eucCN	zh_CN	zh_CN.IBM-eucCN	はい	はい	103P	いいえ	5.2 以前
中国語 (簡体字)	中華人民共和国	GB18030	zh_CN	Zh_CN.GB18030	はい	はい	103P	いいえ	5.2 以前
中国語 (繁体字)	香港特別行政区	BIG5-HKSCS	Zh_HK	Zh_HK.BIG5-HKSCS	はい	はい	467	いいえ	5.2 以前
中国語 (繁体字)	台湾	IBM-eucTW	zh_TW	zh_TW.IBM-eucTW	はい	はい	467	いいえ	5.2 以前
中国語 (繁体字)	台湾	big-5	Zh_TW	Zh_TW.big-5	はい	はい	467	いいえ	5.2 以前
クロアチア語	クロアチア	ISO8859-2	hr_HR	hr_HR.ISO8859-2	いいえ	はい	234	いいえ	5.2 以前
チェコ語	チェコ	ISO8859-2	cs_CZ	cs_CZ.ISO8859-2	はい	はい	243	いいえ	5.2 以前
デンマーク語	デンマーク	ISO8859-15	da_DK.8859-15		いいえ	はい	159	いいえ	5.2 以前
デンマーク語	デンマーク	ISO8859-1	da_DK	da_DK.ISO8859-1	いいえ	はい	159	いいえ	5.2 以前
オランダ語	ベルギー	ISO8859-15	nl_BE.8859-15		いいえ	はい	120	いいえ	5.2 以前
オランダ語	ベルギー	ISO8859-1	nl_BE	nl_BE.ISO8859-1	いいえ	はい	120	いいえ	5.2 以前
オランダ語	ベルギー	IBM-1252	nl_BE.IBM-1252		いいえ	はい	120	いいえ	5.2 以前
オランダ語	オランダ	ISO8859-15	nl_NL.8859-15		いいえ	はい	143	いいえ	5.2 以前
オランダ語	オランダ	ISO8859-1	nl_NL	nl_NL.ISO8859-1	いいえ	はい	143	いいえ	5.2 以前
オランダ語	オランダ	IBM-1252	nl_NL.IBM-1252		いいえ	はい	143	いいえ	5.2 以前
英語	オーストラリア	ISO8859-15	en_AU.8859-15		いいえ	はい	103P	いいえ	5.2 以前
英語	ベルギー	ISO8859-15	en_BE.8859-15		はい	はい	120	いいえ	5.2 以前
英語	カナダ	ISO8859-15	en_CA.8859-15		はい	はい	445	いいえ	5.2 以前
英語	香港特別行政区	ISO8859-15	en_HK.8859-15		はい	はい	168	いいえ	5.2 以前
英語	インド	ISO8859-15	en_IN.8859-15		はい	はい	468	いいえ	5.2 以前
英語	アイルランド	ISO8859-15	en_IE.8859-15		はい	はい	168	いいえ	5.2 以前
英語	ニュージーランド	ISO8859-15	en_NZ.8859-15		はい	はい	103P	いいえ	5.2 以前
英語	フィリピン	ISO8859-15	en_PH.8859-15		はい	はい	103P	いいえ	5.2 以前
英語	シンガポール	ISO8859-15	en_SG.8859-15		はい	はい	168	いいえ	5.2 以前
英語	南アフリカ	ISO8859-15	en_ZA.8859-15		はい	はい	103P	いいえ	5.2 以前
英語	グレートブリテンおよび北部アイルランド連合王国 (英国)	ISO8859-15	en_GB.8859-15		はい	はい	166	いいえ	5.2 以前
英語	グレートブリテンおよび北部アイルランド連合王国 (英国)	ISO8859-1	en_GB	en_GB.ISO8859-1	はい	はい	166	いいえ	5.2 以前
英語	グレートブリテンおよび北部アイルランド連合王国 (英国)	IBM-1252	en_GB.IBM-1252		はい	はい	166	いいえ	5.2 以前
英語	アメリカ合衆国	ISO8859-15	en_US.8859-15		はい	はい	103P	いいえ	5.2 以前
英語	アメリカ合衆国	ISO8859-1	en_US	en_US.ISO8859-1	はい	はい	103P	いいえ	5.2 以前
エストニア語	エストニア	ISO8859-4	et_EE	et_EE.ISO8859-4	いいえ	はい	454	いいえ	5.2 以前
エストニア語	エストニア	IBM-922	Et_EE	Et_EE.IBM-922	いいえ	はい	454	いいえ	5.2 以前
フィンランド語	フィンランド	ISO8859-15	fi_FI.8859-15		いいえ	はい	153	いいえ	5.2 以前
フィンランド語	フィンランド	ISO8859-1	fi_FI	fi_FI.ISO8859-1	いいえ	はい	153	いいえ	5.2 以前
フィンランド語	フィンランド	IBM-1252	fi_FI.IBM-1252		いいえ	はい	153	いいえ	5.2 以前
フランス語	ベルギー	ISO8859-15	fr_BE.8859-15		はい	はい	120	いいえ	5.2 以前
フランス語	ベルギー	ISO8859-1	fr_BE	fr_BE.ISO8859-1	はい	はい	120	いいえ	5.2 以前
フランス語	ベルギー	IBM-1252	fr_BE.IBM-1252		はい	はい	120	いいえ	5.2 以前
フランス語	カナダ	ISO8859-15	fr_CA.8859-15		はい	はい	58	いいえ	5.2 以前
フランス語	カナダ	ISO8859-1	fr_CA	fr_CA.ISO8859-1	はい	はい	58	いいえ	5.2 以前
フランス語	フランス	ISO8859-15	fr_FR.8859-15		はい	はい	120	いいえ	5.2 以前
フランス語	フランス	ISO8859-1	fr_FR	fr_FR.ISO8859-1	はい	はい	120	いいえ	5.2 以前
フランス語	フランス	IBM-1252	fr_FR.IBM-1252		はい	はい	120	いいえ	5.2 以前
フランス語	ルクセンブルグ	ISO8859-15	fr_LU.8859-15		はい	はい	120	いいえ	5.2 以前
フランス語	スイス	ISO8859-15	fr_CH.8859-15		はい	はい	150F	いいえ	5.2 以前
フランス語	スイス	ISO8859-1	fr_CH	fr_CH.ISO8859-1	はい	はい	150F	いいえ	5.2 以前
ドイツ語	オーストリア	ISO8859-15	de_AT.8859-15		はい	はい	129	いいえ	5.2 以前

表 16. AIX での非 *Unicode* エンコード済み言語およびロケール (続き)

言語	地域	コード・セット	デフォルト・ロケール名	別名	翻訳済み	クライアント・サポートの提供	デフォルト **LFT** キーボード **ID**	**CLDR locale?**	**AIX** リリースでの導入
ドイツ語	ドイツ	ISO8859-15	de_DE.8859-15		はい	はい	129	いいえ	5.2 以前
ドイツ語	ドイツ	ISO8859-1	de_DE	de_DE.ISO8859-1	はい	はい	129	いいえ	5.2 以前
ドイツ語	ドイツ	IBM-1252	de_DE.IBM-1252		はい	はい	129	いいえ	5.2 以前
ドイツ語	ルクセンブルグ	ISO8859-15	de_LU.8859-15		はい	はい	150G	いいえ	5.2 以前
ドイツ語	スイス	ISO8859-15	de_CH.8859-15		はい	はい	150G	いいえ	5.2 以前
ドイツ語	スイス	ISO8859-1	de_CH	de_CH.ISO8859-1	はい	はい	150G	いいえ	5.2 以前
ギリシャ語	ギリシャ	ISO8859-7	el_GR	el_GR.ISO8859-7	いいえ	はい	319	いいえ	5.2 以前
ハンガリー語	ハンガリー	ISO8859-2	hu_HU	hu_HU.ISO8859-2	はい	はい	208	いいえ	5.2 以前
アイスランド語	アイスランド	ISO8859-15	is_IS.8859-15		いいえ	はい	197	いいえ	5.2 以前
アイスランド語	アイスランド	ISO8859-1	is_IS	is_IS.ISO8859-1	いいえ	はい	197	いいえ	5.2 以前
インドネシア語	インドネシア	ISO8859-15	id_ID.8859-15		いいえ	はい	103P	いいえ	5.2 以前
イタリア語	イタリア	ISO8859-15	it_IT.8859-15		はい	はい	142	いいえ	5.2 以前
イタリア語	イタリア	ISO8859-1	it_IT	it_IT.ISO8859-1	はい	はい	142	いいえ	5.2 以前
イタリア語	イタリア	IBM-1252	it_IT.IBM-1252		はい	はい	142	いいえ	5.2 以前
イタリア語	スイス	ISO8859-15	it_CH.8859-15		はい	はい	150G	いいえ	5.2 以前
日本語	日本	IBM-eucJP	ja_JP	ja_JP.IBM-eucJP	はい	はい	194	いいえ	5.2 以前
日本語	日本	IBM-943	Ja_JP	Ja_JP.IBM-943	はい	はい	194	いいえ	5.2 以前
韓国語	韓国	IBM-eucKR	ko_KR	ko_KR.IBM-eucKR	はい	はい	413	いいえ	5.2 以前
ラトビア語	ラトビア	ISO8859-4	lv_LV	lv_LV.ISO8859-4	いいえ	はい	455	いいえ	5.2 以前
リトアニア語	リトアニア共和国	ISO8859-4	lt_LT	lt_LT.ISO8859-4	いいえ	はい	456	いいえ	5.2 以前
リトアニア語	リトアニア共和国	IBM-921	Lt_LT	Lt_LT.IBM-921	いいえ	はい	456	いいえ	5.2 以前
マケドニア語	マケドニア	ISO8859-5	mk_MK	mk_MK.ISO8859-5	いいえ	はい	449	いいえ	5.2 以前
マレー語	マレーシア	ISO8859-15	ms_MY.8859-15		いいえ	はい	103P	いいえ	5.2 以前
ポーランド語	ポーランド	ISO8859-2	pl_PL	pl_PL.ISO8859-2	はい	はい	214	いいえ	5.2 以前
ポルトガル語	ブラジル	ISO8859-15	pt_BR.8859-15		はい	はい	275	いいえ	5.2 以前
ポルトガル語	ブラジル	ISO8859-1	pt_BR	pt_BR.ISO8859-1	はい	はい	275	いいえ	5.2 以前
ポルトガル語	ポルトガル	ISO8859-15	pt_PT.8859-15		いいえ	はい	163	いいえ	5.2 以前
ポルトガル語	ポルトガル	ISO8859-1	pt_PT	pt_PT.ISO8859-1	いいえ	はい	163	いいえ	5.2 以前
ポルトガル語	ポルトガル	IBM-1252	pt_PT.IBM-1252		いいえ	はい	163	いいえ	5.2 以前
POSIX	POSIX	ISO8859-1	C	C.ISO8859-1	いいえ	はい	103P	いいえ	5.2 以前
ルーマニア語	ルーマニア	ISO8859-2	ro_RO	ro_RO.ISO8859-2	いいえ	はい	446	いいえ	5.2 以前
ロシア語	ロシア	ISO8859-5	ru_RU	ru_RU.ISO8859-5	はい	はい	441	いいえ	5.2 以前
スロバキア語	スロバキア	ISO8859-2	sk_SK	sk_SK.ISO8859-2	はい	はい	245	いいえ	5.2 以前
スロベニア語	スロベニア	ISO8859-2	sl_SI	sl_SI.ISO8859-2	いいえ	はい	234	いいえ	5.2 以前
スペイン語	アルゼンチン	ISO8859-15	es_AR.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	ボリビア	ISO8859-15	es_BO.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	チリ	ISO8859-15	es_CL.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	コロンビア	ISO8859-15	es_CO.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	コスタリカ	ISO8859-15	es_CR.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	ドミニカ共和国	ISO8859-15	es_DO.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	エクアドル	ISO8859-15	es_EC.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	エルサルバドル共和国	ISO8859-15	es_SV.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	グアテマラ	ISO8859-15	es_GT.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	ホンジュラス	ISO8859-15	es_HN.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	メキシコ	ISO8859-15	es_MX.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	ニカラグア	ISO8859-15	es_NI.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	パナマ	ISO8859-15	es_PA.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	パラグアイ	ISO8859-15	es_PY.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	ペルー	ISO8859-15	es_PE.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	プエルトリコ	ISO8859-15	es_PR.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	スペイン	ISO8859-15	es_ES.8859-15		はい	はい	173	いいえ	5.2 以前
スペイン語	スペイン	ISO8859-1	es_ES	es_ES.ISO8859-1	はい	はい	173	いいえ	5.2 以前
スペイン語	スペイン	IBM-1252	es_ES.IBM-1252		はい	はい	173	いいえ	5.2 以前
スペイン語	アメリカ合衆国	ISO8859-15	es_US.8859-15		はい	はい	103P	いいえ	5.2 以前
スペイン語	ウルグアイ	ISO8859-15	es_UY.8859-15		はい	はい	171	いいえ	5.2 以前
スペイン語	ベネズエラ	ISO8859-15	es_VE.8859-15		はい	はい	171	いいえ	5.2 以前
スウェーデン語	スウェーデン	ISO8859-15	sv_SE.8859-15		いいえ	はい	153	いいえ	5.2 以前
スウェーデン語	スウェーデン	ISO8859-1	sv_SE	sv_SE.ISO8859-1	いいえ	はい	153	いいえ	5.2 以前
タイ語	タイ	TIS-620	th_TH	th_TH.TIS-620	いいえ	はい	191	いいえ	5.2 以前
トルコ語	トルコ	ISO8859-9	tr_TR	tr_TR.ISO8859-9	いいえ	はい	179	いいえ	5.2 以前
ウクライナ語	ウクライナ	IBM-1124	Uk_UA	Uk_UA.IBM-1124	いいえ	はい	465	いいえ	5.2 以前
ベトナム語	ベトナム	IBM-1129	Vi_VN	Vi_VN.IBM-1129	いいえ	はい	461	いいえ	5.2 以前

グローバリゼーションに関する参照情報

ここでは、グローバリゼーションのチェックリストと各国語サポート・サブルーチンに関する情報を記載します。

グローバリゼーションのチェックリスト

グローバリゼーションのチェックリストでは、翻訳および各国語サポートの依存関係についてプログラムを分析する方法を示しています。 このリストを調べることにより、グローバリゼーション機能に関して考慮すべき点があれば、それを判断することができます。 これは、プログラミングとテストの両方で役立ちます。 プログラムが依存する一連のグローバリゼーション項目を特定すれば、テスト・ストラテジーを開発することができます。 これは、すべてのプログラムをテストする共通の方法を容易に実施できるものにします。

グローバリゼーションに関する主な考慮事項は、これまでに示されています。 しかし、このリストはすべてを網羅したものではありません。 リストに入っていないグローバリゼーション関連の質問が、他にもある場合があります。

プログラム操作のチェックリスト

1. プログラムは、直接、間接を問わず、変換可能なメッセージをユーザーに表示しますか。 間接のメッセージの例としては、ライブラリーに保管されたメッセージがあります。

   はい (yes) の場合、
   - これらのメッセージは、メッセージ機能サブルーチンを用いてプログラムから外部化されましたか。
   - このようなメッセージのすべてに、メッセージ・ソース・ファイルを提供しましたか。
   - プログラムはどのロケールのもとで実行されますか?
     - プログラムがロケール環境変数で決められたロケールで実行される場合、次の方法で **setlocale** サブルーチンを呼び出しましたか。

       ```
       setlocale(LC_ALL, "")
       ```

       注: ロケール・カテゴリーは、事前定義の階層順に示すと **LC_ALL**、 **LC_COLLATE**、**LC_CTYPE**、**LC_MESSAGES**、**LC_MONETARY**、**LC_NUMERIC**、および **LC_TIME** となります。
     - ロケール環境変数によって指定されたロケールにメッセージを表示することを除き、プログラムが C ロケールで実行する場合、 **setlocale** サブルーチンを次の方法で呼び出しましたか。

       ```
       setlocale(LC_MESSAGES, "")
       ```
     - **setlocale** サブルーチンを呼び出した後、 **catopen** サブルーチンを次の方法で呼び出しましたか。

       ```
       catopen(catalog_name, NL_CAT_LOCALE)
       ```
     - **catopen** サブルーチンを正しいカタログ名で呼び出しましたか。
     - 変換可能なメッセージの詳細については、 159 ページの『メッセージ機能』を参照してください。
2. プログラムは、テキスト・ストリングを比較しますか。

   はい (yes) の場合、
   - ストリングを比較するのは、等しいかどうかを確認するためだけですか。

はい (yes) の場合、

- **strcmp** あるいは **strncmp** サブルーチンを使用してください。
- **strcoll** あるいは **strxfrm** サブルーチンを使用しないでください。

- 現行ロケールの定義に従ってストリングを比較して、他のものより前にソートするのはどれかを調べますか。

  はい (yes) の場合、

  - **setlocale** サブルーチンを次の方法で呼び出してください。

    ```
    setlocale(LC_ALL, "")
    ```

  - **strcoll**、**strxfrm**、**wcscoll**、または **wcsxfrm** サブルーチンを使用してください。
  - **strxfrm** サブルーチンも **strncmp** サブルーチンも使用しないでください。

3. プログラムでファイルのパス名を構文解析しますか。

   はい (yes) の場合、

   - / (スラッシュ) を検索する場合は、 **strchr** サブルーチンを使用してください。
   - 文字を検索する場合は、ファイル名にマルチバイト文字が含まれている場合があることに注意してください。 このような場合には、 **setlocale** サブルーチンを次の方法で呼び出してから、適切な検索サブルーチンを使用してください。

     ```
     setlocale(LC_ALL, "")
     ```

4. プログラムは、ノード名、ユーザー名、プリンター名、およびキュー名のような、システム名を使用しますか。

   はい (yes) の場合、

   - システム名にはマルチバイト文字を使用しても構いません。
   - マルチバイト文字を見分けるには、まず **setlocale** サブルーチンを次の方法で呼び出してから、ライブラリー内の適切なサブルーチンを使用してください。

     ```
     setlocale(LC_ALL, "")
     ```

5. プログラムは、大文字、小文字、およびアルファベットのような、文字クラス属性を使用しますか。

   はい (yes) の場合、

   - **setlocale** サブルーチンを次の方法で呼び出してください。

     ```
     setlocale(LC_ALL, "")
     ```

   - 文字特性については、前提を決めないでください。 文字特性を判別する場合は、必ずシステム・サブルーチンを使用してください。
   - 文字は単一バイト・コード・セットに限定されていますか。

     はい (yes) の場合、

     - **isalnum**、**isalpha**、**iscntrl**、**isdigit**、**isgraph**、**isprint**、**isspace**、または **isxdigit** のいずれかの **ctype** サブルーチンを使用してください。

       使わない場合は、文字がマルチバイト文字の可能性があります。

     - **iswalnum**、**iswalpha**、**iswcntrl**、**iswdigit**、**iswgraph**、**iswlower**、**iswprint**、**iswpunct**、**iswspace**、**iswupper**、または **iswxdigit** サブルーチンを使用してください。

6. プログラムは文字の大文字小文字を変換しますか。

はい (yes) の場合、

- **setlocale** サブルーチンを次の方法で呼び出してください。

  ```
  setlocale(LC_ALL, "")
  ```

- 文字は単一バイト・コード・セットに限定されていますか。

  はい (yes) の場合、

  – **_tolower**、**_toupper**、**tolower**、または **toupper** などの **conv** サブルーチンを使用してください。

  使わない場合は、文字がマルチバイト文字の可能性があります。

  – **towlower** または **towupper** サブルーチンを使用してください。

7. プログラムは、tty 端末でのカーソル移動をトラッキングしていますか。

   はい (yes) の場合、

   - **setlocale** サブルーチンを次の方法で呼び出してください。

     ```
     setlocale(LC_ALL, "")
     ```

   - 文字の表示列幅の判別が必要になる場合があります。 **wcwidth** または **wcswidth** サブルーチンを使用してください。

8. プログラムは文字入出力を行いますか。

   はい (yes) の場合、

   - **setlocale** サブルーチンを次の方法で呼び出してください。

     ```
     setlocale(LC_ALL, "")
     ```

   - 文字は単一バイト・コード・セットに限定されていますか。

     はい (yes) の場合、

     – 以下のサブルーチン・ファミリーを使用してください。

       - **fgetc**、**getc**、**getchar**、**getw**
       - **fgets**、**gets**
       - **fputc**、**putc**、**putchar**、**putw**
       - **printf**、**scanf**

     いいえ (not) の場合、

     – 以下のサブルーチン・ファミリーを使用してください。

       - **fgetwc**、 **getwc**、**getwchar**
       - **fgetws**、 **getws**
       - **fputwc**、 **putwc**、**putwchar**

9. プログラムは文字の配列を 1 つずつ細かく調べますか。

   はい (yes) の場合、

   - 配列は単一バイト文字に制限されていますか。

     はい (yes) の場合、

     – **setlocale(LC_ALL,"")** は必要ありません。

     – p が単一バイト文字のこの配列を指すポインターである場合は、 p++ を用いてこの配列を 1 つずつ細かく調べてください。

いいえ (not) の場合、

- **setlocale** サブルーチンを次の方法で呼び出してください。

  ```
  setlocale(LC_ALL, "")
  ```

- **mblen** または **wcslen** サブルーチンを使用してください。

10. プログラムは、コード・セット内の文字をエンコードするのに使用する最大バイト数を知る必要がありますか。

    はい (yes) の場合、

    - **setlocale** サブルーチンを次の方法で呼び出してください。

      ```
      setlocale(LC_ALL, "")
      ```

    - **MB_CUR_MAX** マクロを使用してください。

11. プログラムは、日付や時刻の数字をフォーマット設定しますか。

    はい (yes) の場合、

    - **setlocale** サブルーチンを次の方法で呼び出してください。

      ```
      setlocale(LC_ALL, "")
      ```

    - **nl_langinfo** または **localeconv** サブルーチンを使用してロケール固有の情報を入手してください。
    - **strftime** サブルーチンまたは **strptime** サブルーチンを使用してください。

12. プログラムは数字をフォーマット設定しますか。

    はい (yes) の場合、

    - `setlocale`

      サブルーチンを次の方法で呼び出してください。

      ```
      setlocale(LC_ALL, "")
      ```

    - **nl_langinfo** または **localeconv** サブルーチンを使用してロケール固有の情報を入手してください。
    - 必要に応じて、**printf** サブルーチンと **scanf** サブルーチンのペアを使用してください。

13. プログラムは通貨の数量をフォーマット設定しますか。

    はい (yes) の場合、

    - **setlocale** サブルーチンを次の方法で呼び出してください。

      ```
      setlocale(LC_ALL, "")
      ```

    - **nl_langinfo** または **localeconv** サブルーチンを使用してロケール固有の情報を入手してください。
    - **strfmon** サブルーチンを使用して通貨の数字をフォーマット設定してください。

14. プログラムはストリングもしくは文字の検索を行いますか。

    はい (yes) の場合、

    - 単一バイト・テキストの単一バイト文字を探していますか。
      - `setlocale(LC_ALL,"")` は必要ありません。
      - **strchr** のような、標準の **libc** ストリング・サブルーチンを使用してください。
    - 0x00 から 0x3F (固有のコード・ポイント範囲) の文字を探していますか。

- `setlocale(LC_ALL,"")` は必要ありません。
- **strchr**、**strcspn**、**strpbrk**、**strrchr**、**strspn**、**strstr**、**strtok**、および **memchr** サブルーチンのような、標準の **libc** ストリング・サブルーチンを使用してください。
- 0x00 から 0xFF の範囲の文字を探していますか。
- **setlocale** サブルーチンを次の方法で呼び出してください。

  ```
  setlocale(LC_ALL, "")
  ```
- 2 つの方法が使用できます。

  mblen サブルーチンを使用して、マルチバイト文字をスキップします。 次に、単一バイト文字を検出したら、等しいかどうかを確認します。 チェックリスト項目 2 を調べてください。

  または

  検索文字および検索ストリングをワイド文字形式に変換してから、ワイド文字検索サブルーチンを使用してください。

15. プログラムは、正規表現パターン・マッチングを行いますか。

    はい (yes) の場合、
    - **setlocale** サブルーチンを次の方法で呼び出してください。

      ```
      setlocale(LC_ALL, "")
      ```
    - **regcomp**、**regexec**、または **regerror** サブルーチンを使用してください。
16. プログラムは、ユーザーに肯定/否定応答を求めますか。

    はい (yes) の場合、
    - **setlocale** サブルーチンを次の方法で呼び出してください。

      ```
      setlocale(LC_ALL, "")
      ```
    - メッセージ・カタログにプロンプトを入れてください。 **catopen** および **catgets** サブルーチンを使用して、カタログを検索し、プロンプトを表示してください。
    - **rpmatch** サブルーチンを使用してユーザー応答と突き合わせてください。
17. プログラムは、特殊な罫線記号文字を使用しますか。

    はい (yes) の場合、
    - コード・セット固有の罫線記号文字は使用しないでください。
    - 代わりに、**terminfo** ファイルで指定された罫線記号文字と属性を使用してください。
18. プログラムは、ここでは述べない各国固有の処理もしくはロケール固有の処理を行いますか。

    はい (yes) の場合、
    - 各国固有のモジュールを外部化してください。 これらを実行可能プログラムの一部としないでください。
    - このモジュールを、**load** サブルーチンのようなシステム提供のサブルーチンを用いて、実行時にロードしてください。
    - システムにこのような機能がない場合は、それらを静的にリンクしますが、モジュラー方式で提供します。

関連概念:

15 ページの『ロケールの設定』
すべての国際化されたプログラムは、 **setlocale** サブルーチンを使用して、現行ロケールを設定しなければなりません。プロセスでこのサブルーチンを使用すると、ロケール・データベースにアクセスして現行ロケールの変更や照会を行えます。

159 ページの『メッセージ機能』
メッセージを、プログラムが実行時にアクセスすることができるメッセージ・カタログの形式で提供し、プログラムから分けておく必要があります。 この調整により、メッセージをさまざまな言語に翻訳し、ユーザーのロケールに基づいてプログラムで使用することが容易になります。 このタスクに役立つコマンドとサブルーチンが、メッセージ機能に備えられています。

29 ページの『ワイド文字種別サブルーチン』
大多数のワイド文字種別サブルーチンは従来の文字種別サブルーチンに似ていますが、例外としてワイド文字種別サブルーチンは **wint_t** データ型引数として渡される **wchar_t** データ型引数に対して操作を行います。

関連情報:

LC_COLLATE カテゴリー

setlocale サブルーチン

catopen サブルーチン

strcmp サブルーチン

strxfrm サブルーチン

wcscoll サブルーチン

wcsxfrml サブルーチン

strlenl サブルーチン

ctype サブルーチン

iswalnum サブルーチン

サブルーチンのリスト (A から P)

サブルーチンのリスト (Q から Z)

AIXwindows チェックリスト

残りのチェックリスト項目は、AIXwindows システムに固有のものです。

1. プログラムは、X アプリケーションでコード・セットから独立するように、フォント・セット仕様を使用しますか。
2. クライアントは、変換可能なメッセージを表示するのに、ラベル、ボタン、またはその他の出力専用のウィジェットを使用しますか。

   はい (yes) の場合、

   - ***XtSetLanguageProc** サブルーチンを次の方法で呼び出してください。

     ```
     XtSetLanguageProc(NULL, NULL, NULL);
     ```

   - メッセージは、メッセージ・カタログかローカリゼーション・リソース・ファイルのいずれかに入れることができます。 それぞれ、チェックリスト項目の 1 か 20 を参照してください。
   - ウィジェットをコード・セット独立にするには、フォント・セットを使用するフォントを指定してください。
3. クライアントは、ラベル、ボタン、またはテキスト・ウィジェットのテキストを定義するのに、 X リソース・ファイルを使用しますか。

はい (yes) の場合、

- 変換が必要なリソースのすべてを 1 個所に書き込みます。
- テキスト・ストリングにメッセージ・カタログを使用することを考慮してください。
- カラー名は 1 つのエンコードに限定されるので、変換済みのカラー名は使用しないでください。移植可能文字セットでは、移植可能名のみがエンコードされます。
- 言語固有のリソース・ファイルを **/usr/lib/X11/%L/app-defaults/%N** に入れます。ここでは、**%L** は fr_FR のようなロケールの名前であり、 **%N** はクライアントの名前です。

4. キーボード入力は言語によってローカライズされていますか。

   はい (yes) の場合、

   - ***XtSetLanguageProc** サブルーチンを次の方法で呼び出してください。

     ```
     XtSetLanguageProc(NULL, NULL, NULL);
     ```

   - すべてのテキスト入力に **XmText** または **XmTextField** ウィジェットを使用してください。

     XmText ウィジェットの引数には、バイト長ではなく、文字長によって定義されるものもあります。 カーソル位置は、バイト位置ではなく、文字位置で保たれます。

   - ローカリゼーション入力を行うのに、 **XmDrawingArea** ウィジェットを使用していますか。
     - 入力処理をさまざまな言語で行うときは、この入力メソッド・サブルーチンを使用してください。

5. クライアントは、X リソース・ファイルではなくユーザー・ファイルからのローカリゼーション・テキストで構成するリストもしくはラベルを表示しますか。

   はい (yes) の場合、

   - ***XtSetLanguageProc** サブルーチンを次の方法で呼び出してください。

     ```
     XtSetLanguageProc(NULL, NULL, NULL);
     ```

   - **XmStringCreateSimple** サブルーチンを使用して、ローカリゼーション・テキストの **XmString** データ型を作成してください。 **XmStringCreate** サブルーチンは使用できますが、望ましいのは **XmSTRING_DEFAULT_CHARSET** です。
   - ウィジェットをコード・セット独立にするには、フォント・セットを用いてフォントを指定します。 app デフォルト・ファイルのフォント・リソース (例として ***fontList:**) は、小文字形式ではなく大文字かつクラス形成を使用する必要があります (例として ***FontList:**)。こうすると、デスクトップ・スタイル・マネージャーはアプリケーション・フォントの選択に影響を及ぼすことができます。

6. プログラムは、両方向テキストに対して、プレゼンテーション操作 (Xlib 描画、印刷、フォーマット設定、もしくは編集) を行いますか。

   はい (yes) の場合、

   - Xm (Motif) ライブラリーの **XmText** または **XmTextField** を使用してください。 これらのウィジェットは、両方向テキストに使用することができます。
   - Xm ライブラリーが使用できない場合は、レイアウト・サブルーチンを用いて、テキストでの再配列および形成を行ってください。
   - テキストの保管および通信は、暗黙形式 (論理形成) で行います。ビジュアル形式の両方向テキストをサポートするユーティリティーもありますが (例えば、aixterm)、ほとんどの各国語サポート・サブルーチンはビジュアル形式の両方向テキストを処理できません。

上記のすべての項目に対する回答が「いいえ」の場合は、プログラムに各国語サポートの依存関係がない可能性があります。 この場合は、ロケール設定サブルーチン **setlocale** および、カタログ機能サブルーチン **catopen** と **catgets** を必要としない可能性があります。

関連概念:

159 ページの『メッセージ機能』
メッセージを、プログラムが実行時にアクセスすることができるメッセージ・カタログの形式で提供し、プログラムから分けておく必要があります。 この調整により、メッセージをさまざまな言語に翻訳し、ユーザーのロケールに基づいてプログラムで使用することが容易になります。 このタスクに役立つコマンドとサブルーチンが、メッセージ機能に備えられています。

126 ページの『入力メソッド』
グローバリゼーションがベースを提供する国際環境で実行するアプリケーションの場合は、入力メソッドが必要です。入力メソッドは、アプリケーション・プログラミング・インターフェース (API) であり、特定の言語、キーボード、またはコード・セットに依存しないアプリケーションを開発できるようにします。

関連情報:

IMAuxDraw コールバック・サブルーチン

Xm (Motif) ライブラリーでのレイアウト (両方向) サポート

setlocale サブルーチン

catopen サブルーチン

catgets サブルーチン

各国語サポート・サブルーチンのリスト

各国語サポート・サブルーチンは、ロケール固有の情報を処理し、ワイド文字およびマルチバイト文字を操作し、正規表現を使用する場合に使用します。

関連概念:

『ロケール・サブルーチンのリスト』
このセクションでは、ロケール固有のデータの入手と処理を行うために用意されているサブルーチンをリストします。

14 ページの『各国語サポートのサブルーチン』
このセクションでは、プログラマーが国際化されたポータブル・プログラムを開発する際にサブルーチンを使用する場合の指針を示します。可搬性を最大限にするには、標準的な Open Group、ISO/ANSI C、および POSIX 関数を使用してください。

ロケール・サブルーチンのリスト

このセクションでは、ロケール固有のデータの入手と処理を行うために用意されているサブルーチンをリストします。

ロケール固有のデータの入手と処理には、以下のサブルーチンが提供されます。

サブルーチン	説明
localeconv	プログラム・ロケールのロケール依存規則を検索する。
nl_langinfo	プログラム・ロケールの言語または各国ごとの領域に関する情報を戻す。
rpmatch	現行ロケールで応答が肯定か否定かを判別する。
setlocale	プログラムの現行ロケールを変更または照会する。

関連概念:

196 ページの『各国語サポート・サブルーチンのリスト』
各国語サポート・サブルーチンは、ロケール固有の情報を処理し、ワイド文字およびマルチバイト文字を操作し、正規表現を使用する場合に使用します。

時刻と通貨のフォーマット設定サブルーチンのリスト

サブルーチン	説明
strfmon	現行ロケールに従って、通貨ストリングをフォーマット設定する。
strftime	現行ロケールに従って、日付と時刻をフォーマット設定する。
strptime	現行ロケールに従って、文字ストリングを時刻値に変換する。
wcsftime	現行ロケールに従って、日付と時刻をワイド文字ストリングに変換する。

マルチバイト文字サブルーチンのリスト

サブルーチン	説明
mblen	マルチバイト文字の長さを判別する。
mbstowcs	マルチバイト・ストリングからワイド文字ストリングに変換する。
mbtowc	マルチバイト文字をワイド文字に変換する。

ワイド文字サブルーチンのリスト

以下のサブルーチンは、文字を文字コード形式で処理します。

サブルーチン	説明
fgetwc	入力ストリームからワイド文字またはワードを入手する。
fgetws	ストリームからワイド文字ストリングを入手する。
fputwc	ワイド文字またはワードをストリームに書き出す。
fputws	ワイド文字ストリングをストリームに書き出す。
getwc	入力ストリームからワイド文字またはワードを入手する。
getwchar	入力ストリームからワイド文字またはワードを入手する。
getws	ストリームからワイド文字ストリングを入手する。
iswalnum	ワイド文字が英数字かどうかを判別する。
iswalpha	ワイド文字が英字かどうかを判別する。
iswcntrl	ワイド文字が制御文字かどうかを判別する。

サブルーチン	説明
iswctype	ワイド文字の属性を判別する。
iswdigit	ワイド文字が数字かどうかを判別する。
iswgraph	ワイド文字 (「スペース文字」を除く) が印字文字かどうかを判別する。
iswlower	ワイド文字が小文字かどうかを判別する。
iswprint	ワイド文字 (「スペース文字」を含む) が印字文字かどうかを判別する。
iswpunct	ワイド文字が句読文字かどうかを判別する。
iswspace	ワイド文字がブランク・スペースかどうかを判別する。
iswupper	ワイド文字が大文字かどうかを判別する。
iswxdigit	ワイド文字が 16 進数字かどうかを判別する。
putwc	ワイド文字またはワードをストリームに書き出す。
putwchar	ワイド文字またはワードをストリームに書き出す。
putws	ワイド文字ストリングをストリームに書き出す。
strcoll	2 つのストリングを、現行ロケール内のその照合の重みに基づいて比較する。
strxfrm	ストリングをロケール照合値に変換する。
towlower	大文字のワイド文字を小文字のワイド文字に変換する。
towupper	小文字ワイド文字を大文字ワイド文字に変換する。
ungetwc	ワイド文字をストリームにプッシュする。
wcsid	ワイド文字の charsetID を戻す。
wcscat	ワイド文字ストリングを連結する。
wcschr	ワイド文字を検索する。
wcscmp	ワイド文字ストリングを比較する。
wcscoll	ワイド文字ストリングの照合の重みを比較する。
wcscpy	ワイド文字ストリングをコピーする。
wcscspn	ワイド文字ストリングを検索する。
wcslen	ワイド文字ストリング内の文字数を判別する。
wcsncat	指定した数のワイド文字を連結する。
wcsncmp	指定した数のワイド文字を比較する。
wcsncpy	指定した数のワイド文字をコピーする。
wcspbrk	ワイド文字ストリング内で最初に出現したワイド文字を検索する。
wcsrchr	ワイド文字ストリング内で最後に出現したワイド文字を検索する。
wcsspn	ストリングの最初のセグメントのワイド文字数を戻す。
wcstod	ワイド文字ストリングを倍精度浮動小数点値に変換する。
wcstok	ワイド文字ストリングを、一連の独立したワイド文字ストリングに分けます。
wcstol	ワイド文字ストリングを長整数値に変換する。
wcstombs	一連のワイド文字を一連のマルチバイト文字に変換する。
wcstoul	ワイド文字ストリングを符号なし長整数値に変換する。

サブルーチン	説明
wcswcs	ワイド文字ストリング内で最初に出現したワイド文字シーケンスを検索する。
wcswidth	ワイド文字ストリングの表示幅を判別する。
wcsxfrm	ワイド文字ストリングを文字の照合の重みを表す値に変換する。
wctomb	ワイド文字をマルチバイト文字に変換する。
wctype	現行ロケールで定義された有効な属性名のハンドルを入手する。
wcwidth	ワイド文字の表示幅を判別する。

レイアウト・ライブラリー・サブルーチンのリスト

以下のレイアウト・ライブラリー (**libi18n.a**) のサブルーチンで、両方向かつコンテキスト依存のテキストを異なるフォーマットに変換します。

サブルーチン	説明
layout_object_create	レイアウト・コンテキストを初期化する。
layout_object_free	**LayoutObject** 構造体を解放する。
layout_object_editshape	コンテキスト・テキストの形状を編集する。
layout_object_getvalue	**LayoutObject** 構造体の現行のレイアウト値を照会する。
layout_object_setvalue	**LayoutObject** 構造体のレイアウト値を設定する。
layout_object_shapeboxchars	ボックス文字を形成する。
layout_object_transform	**LayoutObject** 構造体の現行のレイアウト値に従って、テキストを変換する。

メッセージ機能サブルーチンのリスト

メッセージ機能は、標準定義のサブルーチンやコマンド、および外部化メッセージ・カタログをサポートするための、製造業者の付加価値拡張機能から構成されます。 これらのカタログは、アプリケーションが必要に応じてメッセージの検索および表示を行うときに使用します。 以下のメッセージ機能サブルーチンは、アプリケーション用のメッセージを入手します。

サブルーチン	説明
catopen	カタログをオープンする。
catgets	カタログからメッセージを入手する。
catclose	カタログをクローズする。
strerror	エラー番号を、現行ロケールに該当するエラー・メッセージ・ストリングにマップする。

コンバーター・サブルーチンのリスト

国際化環境においては、コード・セットからコード・セットへのデータ変換がしばしば必要になります。以下のコンバーター・サブルーチンは、この目的からサポートされています。

サブルーチン	説明
iconv_open	**FromCode** パラメーターによって指定されたコード・セットから **ToCode** パラメーターによって指定されたコード・セットへ文字を変換するときに必要な初期設定を行う。
iconv	**iconv_open** サブルーチンから入手した記述子を使用して、コンバーター関数を呼び出す。
iconv_close	**cd** 変数によって指定された変換ディスクリプターをクローズして、再度使用できるようにする。
ccsidtocs	コード・セットを返す。

入力メソッド・サブルーチンのリスト

入力メソッドは、キー・ストロークを、ロケールが指定したコード・セットの文字ストリングに変換する一組のサブルーチンです。 入力メソッド・サブルーチンには、ロケール固有の入力処理とキーボード制御 (例えば、Ctrl、Alt、Shift、Lock、および Alt-Graphic) の論理が組み込まれています。以下のサブルーチンは、この入力メソッドをサポートします。

サブルーチン	説明
IMAIXMapping	*KeySymbol* と *State* パラメーターのペアをストリングに変換し、そのストリングを指すポインターを戻す。
IMAuxCreate	補助域を作成するよう、アプリケーション・プログラムに指示する。
IMAuxDestroy	補助域の知識を破棄するようにコールバックに通知する。
IMAuxDraw	補助域を描画するよう、アプリケーション・プログラムに指示する。
IMAuxHide	補助域を隠すよう、アプリケーション・プログラムに指示する。
IMBeep	ビープ音を出すよう、アプリケーション・プログラムに指示する。
IMClose	入力メソッドをクローズする。
IMCreate	特定の入力メソッドの 1 つのインスタンスを作成する。
IMDestroy	入力メソッド・インスタンスを破棄する。
IMFilter	入力メソッドが、キーボード・イベントをその内部処理のために使用するかどうかを検査する。
IMFreeKeymap	**IMInitialzieKeymap** サブルーチンによって割り当てられたリソースを解放する。
IMIndicatorDraw	インディケーターを描画するよう、アプリケーション・プログラムに指示する。
IMIndicatorHide	インディケーターを隠すよう、アプリケーション・プログラムに指示する。
IMInitialize	特定言語の入力メソッドを初期化する。
IMInitializeKeymap	特定言語の入力メソッドを初期化する。
IMIoctl	入力メソッドで、さまざまな制御操作または照会操作を行う。
IMLookupString	キーボードのシンボル/状態のペアをユーザー定義のストリングにマップする。
IMProcessAuxiliary	入力メソッドに補助域への入力を通知する。
IMQueryLanguage	指定された言語のサポートの有無を検査する。
IMSimpleMapping	*KeySymbol* と *State* パラメーターのペアをストリングに変換し、そのストリングを指すポインターを戻す。
IMTextCursor	新しいディスプレイのカーソル位置を設定する。
IMTextDraw	次のストリングを描画するよう、アプリケーション・プログラムに求める。
IMTextHide	テキスト域を隠すよう、アプリケーション・プログラムに指示する。

サブルーチン	説明
IMTextStart	事前編集スペースの長さを、アプリケーション・プログラムに通知する。

正規表現サブルーチンのリスト

以下のサブルーチンで、正規表現を処理します。

サブルーチン	説明
regcomp	**regexec** サブルーチンが比較するための正規表現をコンパイルする。
regerror	**regcomp** または **regexec** サブルーチンのいずれかによって戻されたエラー・コードに対応する、現行ロケールに該当するエラー・メッセージを戻す。
regexec	ストリングと、**regcomp** サブルーチンの前のコールからのコンパイル済み正規表現を比較する。
regfree	**regcomp** サブルーチンの前のコールによって割り当てられた、メモリーを解放する。

関連概念:

14 ページの『各国語サポートのサブルーチン』
このセクションでは、プログラマーが国際化されたポータブル・プログラムを開発する際にサブルーチンを使用する場合の指針を示します。可搬性を最大限にするには、標準的な Open Group、ISO/ANSI C、および POSIX 関数を使用してください。

文字マップ

このセクションには、文字マップのテキスト表記が含まれています。

このセクションには、 50 ページの『各国語サポートのコード・セット』で説明した文字マップのテキスト表記が含まれています。

ISO コード・セット

このセクションでは、ISO コード・セットについて説明します。

ISO8859–1

以下の情報では、ISO8859–1 用のコード・セットについて記述しています。

表 *17. ISO8859–1* コード・セット

シンボル名	**16** 進値
No break space (改行されないスペース)	A0
Inverted exclamation mark (反転感嘆符)	A1
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Currency sign (通貨記号)	A4
Yen sign (円記号)	A5
Broken bar (破線)	A6
Section sign (セクション記号)	A7
Diaeresis (分音符号)	A8
Copyright sign (著作権記号)	A9
Feminine ordinal indicator (女性を示す標識)	AA

表 *17. ISO8859–1* コード・セット *(*続き*)*

シンボル名	**16** 進値
Left-pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Registered sign (登録商標)	AE
Macron (長音符号)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Acute accent (揚音アクセント)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Cedilla (セディーユ)	B8
Superscript one (上付きの 1)	B9
Masculine ordinal indicator (男性を示す標識)	BA
Right-pointing double angle quotation mark (右の二重角引用符)	BB
Vulgar fraction one quarter (分数 4 分の 1)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Vulgar fraction three quarters (分数 4 分の 3)	BE
Inverted question mark (反転疑問符)	BF
Latin capital letter A with grave (抑音付きラテン大文字 A)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2
Latin capital letter A with tilde (波形記号付きラテン大文字 A)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter AE (ラテン大文字 AE)	C6
Latin capital letter C with cedilla (セディーユ付きラテン大文字 C)	C7
Latin capital letter E with grave (抑音付きラテン大文字 E)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter E with circumflex (曲折アクセント記号付きラテン大文字 E)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB

表 17. *ISO8859–1* コード・セット (続き)

シンボル名	**16** 進値
Latin capital letter I with grave (抑音付きラテン大文字 I)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter I with diaeresis (分音符号付きラテン大文字 I)	CF
Latin capital letter eth (ラテン大文字 eth)	D0
Latin capital letter N with tilde (波形記号付きラテン大文字 N)	D1
Latin capital letter O with grave (抑音付きラテン大文字 O)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with tilde (波形記号付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7
Latin capital letter O with stroke (斜線付きラテン大文字 O)	D8
Latin capital letter U with grave (抑音付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with circumflex (曲折アクセント記号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter Y with acute (揚音付きラテン大文字 Y)	DD
Latin capital letter thorn (ラテン大文字 thorn)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter a with grave (抑音付きラテン小文字 a)	E0
Latin small letter a with acute (揚音付きラテン小文字 a)	E1
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with tilde (波形符号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter ae (ラテン小文字 ae)	E6

表 *17. ISO8859–1* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin small letter c with cedilla (セディーユ付きラテン小文字 c)	E7
Latin small letter e with grave (抑音付きラテン小文字 e)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with circumflex (曲折アクセント記号付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB
Latin small letter i with grave (抑音付きラテン小文字 i)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter i with diaeresis (分音符号付きラテン小文字 i)	EF
Latin small letter eth (ラテン小文字 eth)	F0
Latin small letter n with tilde (波形符号付きラテン小文字 n)	F1
Latin small letter o with grave (抑音付きラテン小文字 o)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with tilde (波形符号付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter o with stroke (斜線付きラテン小文字 o)	F8
Latin small letter u with grave (抑音付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with circumflex (曲折アクセント記号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter y with acute (揚音付きラテン小文字 y)	FD
Latin small letter thorn (ラテン小文字 thorn)	FE
Latin small letter y with diaeresis (分音符号付きラテン小文字 y)	FF

ISO8859–2

以下の情報では、ISO8859–2 用のコード・セットについて記述しています。

表 *18. ISO8859–2* コード・セット

シンボル名	**16** 進値
No break space (改行されないスペース)	A0
Latin capital letter A with ogonek (ogonek 付きラテン大文字 A)	A1
Bleve	A2
Capital letter L with stroke (斜線付きラテン大文字 L)	A3
Currency sign (通貨記号)	A4
Latin capital letter L with caron (caron 付きラテン大文字 L)	A5
Latin capital letter S with acute (揚音付きラテン大文字 S)	A6
Section sign (セクション記号)	A7
Diaeresis (分音符号)	A8
Latin capital letter S with caron (caron 付きラテン大文字 S)	A9
Latin capital letter S with cedilla (セディーユ付きラテン大文字 S)	AA
Latin capital letter T with caron (caron 付きラテン大文字 T)	AB
Latin capital letter Z with acute (揚音付きラテン大文字 Z)	AC
Soft hyphen (ソフト・ハイフン)	AD
Latin capital letter Z with caron (caron 付きラテン大文字 Z)	AE
Latin capital letter Z with dot above (上部ドット付きラテン大文字 Z)	AF
Degree sign (度記号)	B0
Latin small letter A with ogenek (ogenek 付きラテン小文字 a)	B1
Ogenek	B2
Latin small letter l with stroke (斜線付きラテン小文字 l)	B3
Acute accent (揚音アクセント)	B4
Latin small letter l with caron (caron 付きラテン小文字 l)	B5
Latin small letter s with acute (揚音付きラテン小文字 s)	B6
Caron	B7
Cedilla (セディーユ)	B8
Latin small letter s with caron (caron 付きラテン小文字 s)	B9
Latin small letter s with cedilla (セディーユ付きラテン小文字 s)	BA
Latin small letter t with caron (caron 付きラテン小文字 t)	BB
Latin small letter z with acute (揚音付きラテン小文字 z)	BC
Double acute accent (二重揚音アクセント)	BD

表 *18. ISO8859–2* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin small letter z with caron (caron 付きラテン小文字 z)	BE
Latin small letter z with dot above (上部ドット付きラテン小文字 z)	BF
Latin capital letter R with acute (揚音付きラテン大文字 R)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2
Latin capital letter A with breve (短音記号付きラテン大文字 A)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter L with acute (揚音付きラテン大文字 L)	C5
Latin capital letter C with acute (揚音付きラテン大文字 C)	C6
Latin capital letter C with cedilla (セディーユ付きラテン大文字 C)	C7
Latin capital letter C with caron (caron 付きラテン大文字 C)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter E with ogonek (ogonek 付きラテン大文字 E)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB
Latin capital letter E with caron (caron 付きラテン大文字 E)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter D with caron (caron 付きラテン大文字 D)	CF
Latin capital letter D with stroke (斜線付きラテン大文字 D)	D0
Latin capital letter N with acute (揚音付きラテン大文字 N)	D1
Latin capital letter N with caron (caron 付きラテン大文字 N)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with double acute (二重揚音付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6

表 *18. ISO8859–2* コード・セット *(*続き*)*

シンボル名	**16** 進値
Multiplication sign (乗算記号)	D7
Latin capital letter R with caron (caron 付きラテン大文字 R)	D8
Latin capital letter U with ring above (上部リング付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with double acute (二重揚音付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter Y with acute (揚音付きラテン大文字 Y)	DD
Latin capital letter T with cedilla (セディーユ付きラテン大文字 T)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter r with acute (揚音付きラテン小文字 r)	E0
Latin small letter a with acute (揚音付きラテン小文字 a)	E1
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with breve (短音記号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter l with acute (揚音付きラテン小文字 l)	E5
Latin small letter c with acute (揚音付きラテン小文字 c)	E6
Latin small letter c with cedilla (セディーユ付きラテン小文字 c)	E7
Latin small letter c with caron (caron 付きラテン小文字 c)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with ogonek (ogonek 付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB
Latin small letter e with caron (caron 付きラテン小文字 e)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter d with caron (caron 付きラテン小文字 d)	EF

表 *18. ISO8859–2* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin small letter d with stroke (斜線付きラテン小文字 d)	F0
Latin small letter n with acute (揚音付きラテン小文字 n)	F1
Latin small letter n with caron (caron 付きラテン小文字 n)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with double acute (二重揚音付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter r with caron (caron 付きラテン小文字 r)	F8
Latin small letter u with ring above (上部リング付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with double acute (二重揚音付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter y with acute (揚音付きラテン小文字 y)	FD
Latin small letter t with cedilla (セディーユ付きラテン小文字 t)	FE
Dot above (上部ドット)	FF

ISO8859–4

以下の情報では、ISO8859–4 用のコード・セットについて記述しています。

表 *19. ISO8859–4* コード・セット

シンボル名	**16** 進値
No break space (改行されないスペース)	A0
Latin capital letter A with ogonek (ogonek 付きラテン大文字 A)	A1
Latin small letter kra (ラテン小文字 kra)	A2
Latin capital letter R with cedilla (セディーユ付きラテン大文字 R)	A3
Currency sign (通貨記号)	A4
Latin capital letter I with tilde (波形記号付きラテン大文字 I)	A5
Latin capital letter L with cedilla (セディーユ付きラテン大文字 L)	A6
Section sign (セクション記号)	A7
Diaeresis (分音符号)	A8
Latin capital letter S with caron (caron 付きラテン大文字 S)	A9
Latin capital letter E with macron (長音符号付きラテン大文字 E)	AA

表 19. *ISO8859–4* コード・セット (続き)

シンボル名	**16** 進値
Latin capital letter G with cedilla (セディーユ付きラテン大文字 G)	AB
Latin capital letter T with stroke (斜線付きラテン大文字 T)	AC
Soft hyphen (ソフト・ハイフン)	AD
Latin capital letter Z with caron (caron 付きラテン大文字 Z)	AE
Macron (長音符号)	AF
Degree sign (度記号)	B0
Latin small letter a with ogonek (ogonek 付きラテン小文字 a)	B1
Ogonek	B2
Latin small letter r with cedilla (セディーユ付きラテン小文字 r)	B3
Acute accent (揚音アクセント)	B4
Latin small letter i with tilde (波形記号付きラテン小文字 i)	B5
Latin small letter l with cedilla (セディーユ付きラテン小文字 l)	B6
Caron	B7
Cedilla (セディーユ)	B8
Latin small letter s with caron (caron 付きラテン小文字 s)	B9
Latin small letter e with macron (長音符号付きラテン小文字 e)	BA
Latin small letter g with cedilla (セディーユ付きラテン小文字 g)	BB
Latin small letter t with stroke (斜線付きラテン小文字 t)	BC
Latin capital letter eng (ラテン大文字 eng)	BD
Latin small letter z with caron (caron 付きラテン小文字 z)	BE
Latin small letter eng (ラテン小文字 eng)	BF
Latin capital letter A with macron (長音符号付きラテン大文字 A)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2
Latin capital letter A with tilde (波形記号付きラテン大文字 A)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter ae (ラテン大文字 ae)	C6
Latin capital letter I with ogonek (ogonek 付きラテン大文字 I)	C7
Latin capital letter C with caron (caron 付きラテン大文字 C)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter E with ogonek (ogonek 付きラテン大文字 E)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB
Latin capital letter E with dot above (上部ドット付きラテン大文字 E)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter I with macron (長音符号付きラテン大文字 I)	CF
Latin capital letter D with stroke (斜線付きラテン大文字 D)	D0
Latin capital letter N with cedilla (セディーユ付きラテン大文字 N)	D1
Latin capital letter O with macron (長音符号付きラテン大文字 O)	D2
Latin capital letter K with cedilla (セディーユ付きラテン大文字 K)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with tilde (波形記号付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7

表 *19. ISO8859–4* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin capital letter O with stroke (斜線付きラテン大文字 O)	D8
Latin capital letter U with ogonek (ogonek 付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with circumflex (曲折アクセント記号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter U with tilde (波形記号付きラテン大文字 U)	DD
Latin capital letter U with macron (長音符号付きラテン大文字 U)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter a with macron (長音符号付きラテン小文字 a)	E0
Latin small letter a with acute (揚音付きラテン小文字 a)	E1
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with tilde (波形符号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter ae (ラテン小文字 ae)	E6
Latin small letter i with ogonek (ogonek 付きラテン小文字 i)	E7
Latin small letter c with caron (caron 付きラテン小文字 c)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with ogonek (ogonek 付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB
Latin small letter e with dot above (上部ドット付きラテン小文字 e)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter i with macron (長音符号付きラテン小文字 i)	EF
Latin small letter d with stroke (斜線付きラテン小文字 d)	F0
Latin small letter n with cedilla (セディーユ付きラテン小文字 n)	F1
Latin small letter o with macron (長音符号付きラテン小文字 o)	F2
Latin small letter k with cedilla (セディーユ付きラテン小文字 k)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with tilde (波形符号付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter o with stroke (斜線付きラテン小文字 o)	F8
Latin small letter u with ogonek (ogonek 付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with circumflex (曲折アクセント記号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter u with tilde (波形符号付きラテン小文字 u)	FD
Latin small letter u with macron (長音符号付きラテン小文字 u)	FE
Dot above (上部ドット)	FF

ISO8859–5

以下の情報では、ISO8859–5 用のコード・セットについて記述しています。

表 *20. ISO8859–5* コード・セット

シンボル名	**16** 進値
No break space (改行されないスペース)	A0
Cyrillic capital letter io (キリル大文字 io)	A1
Cyrillic capital letter dje (キリル大文字 dje)	A2
Cyrillic capital letter gje (キリル大文字 gje)	A3
Cyrillic capital letter ukrainian ie (キリル大文字 ウクライナ ie)	A4
Cyrillic capital letter dze (キリル大文字 dze)	A5
Cyrillic capital letter byelorussian-ukrainian I (キリル大文字 ベラルーシ - ウクライナ I)	A6
Cyrillic capital letter yi (キリル大文字 yi)	A7
Cyrillic capital letter je (キリル大文字 je)	A8
Cyrillic capital letter lje (キリル大文字 lje)	A9
Cyrillic capital letter nje (キリル大文字 nje)	AA
Cyrillic capital letter tshe (キリル大文字 tshe)	AB
Cyrillic capital letter kje (キリル大文字 kje)	AC
Soft hyphen (ソフト・ハイフン)	AD
Cyrillic capital letter short U (キリル大文字 ショート U)	AE
Cyrillic capital letter dzhe (キリル大文字 dzhe)	AF
Cyrillic capital letter A (キリル大文字 A)	B0
Cyrillic capital letter be (キリル大文字 be)	B1
Cyrillic capital letter ve (キリル大文字 ve)	B2
Cyrillic capital letter ghe (キリル大文字 ghe)	B3
Cyrillic capital letter de (キリル大文字 de)	B4
Cyrillic capital letter ie (キリル大文字 ie)	B5
Cyrillic capital letter zhe (キリル大文字 zhe)	B6
Cyrillic capital letter ze (キリル大文字 ze)	B7
Cyrillic capital letter I (キリル大文字 I)	B8
Cyrillic capital letter short I (キリル大文字 ショート I)	B9
Cyrillic capital letter ka (キリル大文字 ka)	BA
Cyrillic capital letter el (キリル大文字 el)	BB
Cyrillic capital letter em (キリル大文字 em)	BC
Cyrillic capital letter en (キリル大文字 en)	BD
Cyrillic capital letter O (キリル大文字 O)	BE
Cyrillic capital letter pe (キリル大文字 pe)	BF
Cyrillic capital letter er (キリル大文字 er)	C0
Cyrillic capital letter es (キリル大文字 es)	C1
Cyrillic capital letter te (キリル大文字 te)	C2
Cyrillic capital letter U (キリル大文字 U)	C3
Cyrillic capital letter ef (キリル大文字 ef)	C4
Cyrillic capital letter ha (キリル大文字 ha)	C5
Cyrillic capital letter tse (キリル大文字 tse)	C6
Cyrillic capital letter che (キリル大文字 che)	C7
Cyrillic capital letter sha (キリル大文字 sha)	C8
Cyrillic capital letter shcha (キリル大文字 shcha)	C9

表 *20. ISO8859–5* コード・セット *(*続き*)*

シンボル名	**16** 進値
Cyrillic capital letter hard sign (キリル大文字硬音記号)	CA
Cyrillic capital letter yeru (キリル大文字 yeru)	CB
Cyrillic capital letter soft sign (キリル大文字軟音記号)	CC
Cyrillic capital letter E (キリル大文字 E)	CD
Cyrillic capital letter tu (キリル大文字 tu)	CE
Cyrillic capital letter ya (キリル大文字 ya)	CF
Cyrillic small letter a (キリル小文字 a)	D0
Cyrillic small letter be (キリル小文字 be)	D1
Cyrillic small letter ve (キリル小文字 ve)	D2
Cyrillic small letter ghe (キリル小文字 ghe)	D3
Cyrillic small letter de (キリル小文字 de)	D4
Cyrillic small letter ie (キリル小文字 ie)	D5
Cyrillic small letter zhe (キリル小文字 zhe)	D6
Cyrillic small letter ze (キリル小文字 ze)	D7
Cyrillic small letter i (キリル小文字 i)	D8
Cyrillic small letter short I (キリル小文字 ショート I)	D9
Cyrillic small letter ka (キリル小文字 ka)	DA
Cyrillic small letter el (キリル小文字 el)	DB
Cyrillic small letter em (キリル小文字 em)	DC
Cyrillic small letter en (キリル小文字 en)	DD
Cyrillic small letter o (キリル小文字 o)	DE
Cyrillic small letter pe (キリル小文字 pe)	DF
Cyrillic small letter er (キリル小文字 er)	E0
Cyrillic small letter es (キリル小文字 es)	E1
Cyrillic small letter te (キリル小文字 te)	E2
Cyrillic small letter u (キリル小文字 u)	E3
Cyrillic small letter ef (キリル小文字 ef)	E4
Cyrillic small letter ha (キリル小文字 ha)	E5
Cyrillic small letter tse (キリル小文字 tse)	E6
Cyrillic small letter che (キリル小文字 che)	E7
Cyrillic small letter sha (キリル小文字 sha)	E8
Cyrillic small letter shcha (キリル小文字 shcha)	E9
Cyrillic small letter hard sign (キリル小文字硬音記号)	EA
Cyrillic small letter yeru (キリル小文字 yeru)	EB
Cyrillic small letter soft sign (キリル小文字軟音記号)	EC
Cyrillic small letter e (キリル小文字 e)	ED
Cyrillic small letter yu (キリル小文字 yu)	EE
Cyrillic small letter ta (キリル小文字 ta)	EF
Numero sign (ナンバー記号)	F0
Cyrillic small letter io (キリル小文字 io)	F1
Cyrillic small letter dje (キリル小文字 dje)	F2
Cyrillic small letter gje (キリル小文字 gje)	F3

表 *20. ISO8859–5* コード・セット *(*続き*)*

シンボル名	**16** 進値
Cyrillic small letter ukrainian ie (キリル小文字 ウクライナ ie)	F4
Cyrillic small letter dze (キリル小文字 dze)	F5
Cyrillic small letter byelorussian-ukrainian I (キリル小文字 ベラルーシ - ウクライナ I)	F6
Cyrillic small letter yi (キリル小文字 yi)	F7
Cyrillic small letter je (キリル小文字 je)	F8
Cyrillic small letter lje (キリル小文字 lje)	F9
Cyrillic small letter nje (キリル小文字 nje)	FA
Cyrillic small letter tshe (キリル小文字 tshe)	FB
Cyrillic small letter kje (キリル小文字 kje)	FC
Selection sign (選択記号)	FD
Cyrillic small letter short u (キリル小文字 ショート u)	FE
Cyrillic small letter dzhe (キリル小文字 dzhe)	FF

ISO8859–6

以下の情報では、ISO8859–6 用のコード・セットについて記述しています。

表 *21. ISO8859–6* コード・セット

シンボル名	**16** 進値
No-break space (改行されないスペース)	A0
Currency sign (通貨記号)	A4
Arabic comma (アラビア文字 コンマ)	AC
Soft hyphen (ソフト・ハイフン)	AD
Arabic semicolon (アラビア文字 セミコロン)	BB
Arabic question mark (アラビア文字 疑問符)	BF
Arabic letter hamza (アラビア文字 hamza)	C1
Arabic letter alef with madda above (上部 madda 付きアラビア文字 alef)	C2
Arabic letter alef with hamza above (上部 hamza 付きアラビア文字 alef)	C3
Arabic letter waw with hamza above (上部 hamza 付きアラビア文字 waw)	C4
Arabic letter alef with hamza below (下部 hamza 付きアラビア文字 alef)	C5
Arabic letter yeh with hamza above (上部 hamza 付きアラビア文字 yeh)	C6
Arabic letter alef (アラビア文字 alef)	C7
Arabic letter beh (アラビア文字 beh)	C8
Arabic letter teh marbuta (アラビア文字 teh marbuta)	C9
Arabic letter teh (アラビア文字 teh)	CA
Arabic letter theh (アラビア文字 theh)	CB
Arabic letter jeem (アラビア文字 jeem)	CC
Arabic letter hah (アラビア文字 hah)	CD

表 21. *ISO8859–6* コード・セット (続き)

シンボル名	**16** 進値
Arabic letter khah (アラビア文字 khah)	CE
Arabic letter dal (アラビア文字 dal)	CF
Arabic letter thal (アラビア文字 thal)	D0
Arabic letter reh (アラビア文字 reh)	D1
Arabic letter zain (アラビア文字 zain)	D2
Arabic letter seen (アラビア文字 seen)	D3
Arabic letter sheen (アラビア文字 sheen)	D4
Arabic letter sad (アラビア文字 sad)	D5
Arabic letter dad (アラビア文字 dad)	D6
Arabic letter tah (アラビア文字 tah)	D7
Arabic letter zah (アラビア文字 zah)	D8
Arabic letter ain (アラビア文字 ain)	D9
Arabic letter ghain (アラビア文字 ghain)	DA
Arabic letter tatweel (アラビア文字 tatweel)	E0
Arabic letter feh (アラビア文字 feh)	E1
Arabic letter qaf (アラビア文字 qaf)	E2
Arabic letter kaf (アラビア文字 kaf)	E3
Arabic letter lam (アラビア文字 lam)	E4
Arabic letter meem (アラビア文字 meem)	E5
Arabic letter noon (アラビア文字 noon)	E6
Arabic letter heh (アラビア文字 heh)	E7
Arabic letter waw (アラビア文字 waw)	E8
Arabic letter alef maksura (アラビア文字 alef maksura)	E9
Arabic letter yeh (アラビア文字 yeh)	EA
Arabic letter fathatan (アラビア文字 fathatan)	EB
Arabic letter dammatan (アラビア文字 dammatan)	EC
Arabic letter kasratan (アラビア文字 kasratan)	ED
Arabic letter fatha (アラビア文字 fatha)	EE
Arabic letter damma (アラビア文字 damma)	EF
Arabic letter kasra (アラビア文字 kasra)	F0
Arabic letter shadda (アラビア文字 shadda)	F1
Arabic letter sukun (アラビア文字 sukun)	F2

ISO8859–7

以下の情報では、ISO8859–7 用のコード・セットについて記述しています。

表 22. *ISO8859–7*

シンボル名	**16** 進値
No break space (改行されないスペース)	A0
Left single quotation mark (左の引用符)	A1
Right single quotation mark (右の引用符)	A2
Pound sign (ポンド記号)	A3
Euro sign (ユーロ記号)	A4
Broken bar (破線)	A6
Section sign (セクション記号)	A7
Diaeresis (分音符号)	A8
Copyright sign (著作権記号)	A9
Left-pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Horizontal bar (水平バー)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Greek tonos (ギリシャ文字 tonos)	B4
Greek dialytika tonos (ギリシャ文字 dialytika tonos)	B5
Greek capital letter alpha with tonos (tonos 付きギリシャ大文字 アルファ)	B6
Middle dot (中黒)	B7
Greek capital letter epsilon with tonos (tonos 付きギリシャ大文字 イプシロン)	B8
Greek capital letter eta with tonos (tonos 付きギリシャ大文字 イータ)	B9
Greek capital letter iota with tonos (tonos 付きギリシャ大文字 イオタ)	BA
Right-pointing double angle quotation mark (右の二重角引用符)	BB
Greek capital letter omicron with tonos (tonos 付きギリシャ大文字 オミクロン)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Greek capital letter upsilon with tonos (tonos 付きギリシャ大文字 ユプシロン)	BE
Greek capital letter omega with tonos (tonos 付きギリシャ大文字 オメガ)	BF
Greek small letter iota with dialytika and tonos (dialytika と tonos 付きギリシャ小文字 イオタ)	C0
Greek capital letter alpha (ギリシャ大文字 アルファ)	C1
Greek capital letter beta (ギリシャ大文字 ベータ)	C2
Greek capital letter gamma (ギリシャ大文字 ガンマ)	C3
Greek capital letter delta (ギリシャ大文字 デルタ)	C4
Greek capital letter epsilon (ギリシャ大文字 イプシロン)	C5
Greek capital letter zeta (ギリシャ大文字 ゼータ)	C6

表 22. *ISO8859–7 (続き)*

シンボル名	**16** 進値
Greek capital letter eta (ギリシャ大文字 イータ)	C7
Greek capital letter theta (ギリシャ大文字 シータ)	C8
Greek capital letter iota (ギリシャ大文字 イオタ)	C9
Greek capital letter kappa (ギリシャ大文字 カッパ)	CA
Greek capital letter lambda (ギリシャ大文字 ラムダ)	CB
Greek capital letter mu (ギリシャ大文字 ミュー)	CC
Greek capital letter nu (ギリシャ大文字 ニュー)	CD
Greek capital letter xi (ギリシャ大文字 クシー)	CE
Greek capital letter omicron (ギリシャ大文字 オミクロン)	CF
Greek capital letter pi (ギリシャ大文字 パイ)	D0
Greek capital letter rho (ギリシャ大文字 ロー)	D1
Greek capital letter sigma (ギリシャ大文字 シグマ)	D3
Greek capital letter tau (ギリシャ大文字 タウ)	D4
Greek capital letter upsilon (ギリシャ大文字 ユプシロン)	D5
Greek capital letter phi (ギリシャ大文字 ファイ)	D6
Greek capital letter chi (ギリシャ大文字 カイ)	D7
Greek capital letter psi (ギリシャ大文字 プシー)	D8
Greek capital letter omega (ギリシャ大文字 オメガ)	D9
Greek capital letter iota with dialytika (dialytika 付きギリシャ大文字 イオタ)	DA
Greek capital letter upsilon with dialytika (dialytika 付きギリシャ大文字 ユプシロン)	DB
Greek small letter alpha with tonos (tonos 付きギリシャ小文字 アルファ)	DC
Greek small letter epsilon with tonos (tonos 付きギリシャ小文字 イプシロン)	DD
Greek small letter eta with tonos (tonos 付きギリシャ小文字 イータ)	DE
Greek small letter iota with tonos (tonos 付きギリシャ小文字 イオタ)	DF
Greek small letter upsilon with dialytika and tonos (dialytika と tonos 付きギリシャ小文字 ユプシロン)	E0
Greek small letter alpha (ギリシャ小文字 アルファ)	E1
Greek small letter beta (ギリシャ小文字 ベータ)	E2
Greek small letter gamma (ギリシャ小文字 ガンマ)	E3
Greek small letter delta (ギリシャ小文字 デルタ)	E4
Greek small letter epsilon (ギリシャ小文字 イプシロン)	E5
Greek small letter zeta (ギリシャ小文字 ゼータ)	E6
Greek small letter eta (ギリシャ小文字 イータ)	E7
Greek small letter theta (ギリシャ小文字 シータ)	E8
Greek small letter iota (ギリシャ小文字 イオタ)	E9
Greek small letter kappa (ギリシャ小文字 カッパ)	EA
Greek small letter lambda (ギリシャ小文字 ラムダ)	EB
Greek small letter mu (ギリシャ小文字 ミュー)	EC

表 22. *ISO8859–7 (続き)*

シンボル名	**16** 進値
Greek small letter nu (ギリシャ小文字 ニュー)	ED
Greek small letter xi (ギリシャ小文字 クシー)	EE
Greek small letter omicron (ギリシャ小文字 オミクロン)	EF
Greek small letter pi (ギリシャ小文字 パイ)	F0
Greek small letter rho (ギリシャ小文字 ロー)	F1
Greek small letter final sigma (ギリシャ小文字 語尾のシグマ)	F2
Greek small letter sigma (ギリシャ小文字 シグマ)	F3
Greek small letter tau (ギリシャ小文字 タウ)	F4
Greek small letter upsilon (ギリシャ小文字 ユプシロン)	F5
Greek small letter phi (ギリシャ小文字 ファイ)	F6
Greek small letter chi (ギリシャ小文字 カイ)	F7
Greek small letter psi (ギリシャ小文字 プシー)	F8
Greek small letter omega (ギリシャ小文字 オメガ)	F9
Greek small letter iota with dialytika (dialytika 付きギリシャ小文字 イオタ)	FA
Greek small letter upsilon with dialytika (dialytika 付きギリシャ小文字 ユプシロン)	FB
Greek small letter omicron with tonos (tonos 付きギリシャ小文字 オミクロン)	FC
Greek small letter upsilon with tonos (tonos 付きギリシャ小文字 ユプシロン)	FD
Greek small letter omega with tonos (tonos 付きギリシャ小文字 オメガ)	FE

ISO8859–8

以下の情報では、ISO8859–8 用のコード・セットについて記述しています。

表 23. *ISO8859–8*

シンボル名	**16** 進値
No-break space (改行されないスペース)	A0
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Currency sign (通貨記号)	A4
Yen sign (円記号)	A5
Broken bar (破線)	A6
Section sign (セクション記号)	A7
Diaeresis (分音符号)	A8
Copyright sign (著作権記号)	A9
Multiplication sign (乗算記号)	AA
Left-pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD

表 23. *ISO8859–8 (続き)*

シンボル名	**16** 進値
Registered sign (登録商標)	AE
Overline (上線)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Acute accent (揚音アクセント)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Cedilla (セディーユ)	B8
Superscript one (上付きの 1)	B9
Division sign (除算記号)	BA
Right-pointing double angle quotation mark (右の二重角引用符)	BB
Vulgar fraction one quarter (分数 4 分の 1)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Vulgar fraction three quarters (分数 4 分の 3)	BE
Double low line (二重下線)	DF
Hebrew letter alef (ヘブライ文字 alef)	EO
Hebrew letter bet (ヘブライ文字 bet)	E1
Hebrew letter gimel (ヘブライ文字 gimel)	E2
Hebrew letter dalet (ヘブライ文字 dalet)	E3
Hebrew letter he (ヘブライ文字 he)	E4
Hebrew letter vav (ヘブライ文字 vav)	E5
Hebrew letter zayin (ヘブライ文字 zayin)	E6
Hebrew letter het (ヘブライ文字 het)	E7
Hebrew letter tet (ヘブライ文字 tet)	E8
Hebrew letter yod (ヘブライ文字 yod)	E9
Hebrew letter final kaf (ヘブライ文字 語尾の kaf)	EA
Hebrew letter kaf (ヘブライ文字 kaf)	EB
Hebrew letter lamed (ヘブライ文字 lamed)	EC
Hebrew letter final mem (ヘブライ文字 語尾の mem)	ED
Hebrew letter mem (ヘブライ文字 mem)	EE
Hebrew letter final nun (ヘブライ文字 語尾の nun)	EF
Hebrew letter nun (ヘブライ文字 nun)	F0
Hebrew letter samekh (ヘブライ文字 samekh)	F1
Hebrew letter ayin (ヘブライ文字 ayin)	F2
Hebrew letter final pe (ヘブライ文字 語尾の pe)	F3
Hebrew letter pe (ヘブライ文字 pe)	F4
Hebrew letter final tsadi (ヘブライ文字 語尾の tsadi)	F5
Hebrew letter tsadi (ヘブライ文字 tsadi)	F6
Hebrew letter qof (ヘブライ文字 qof)	F7
Hebrew letter resh (ヘブライ文字 resh)	F8
Hebrew letter shin (ヘブライ文字 shin)	F9

表 23. *ISO8859–8 (続き)*

シンボル名	**16** 進値
Hebrew letter tav (ヘブライ文字 tav)	FA

ISO8859–9

以下の情報では、ISO8859–9 用のコード・セットについて記述しています。

表 24. *ISO8859–9* コード・セット

シンボル名	**16** 進値
No-break space (改行されないスペース)	A0
Inverted exclamation mark (反転感嘆符)	A1
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Currency sign (通貨記号)	A4
Yen sign (円記号)	A5
Broken bar (破線)	A6
Section sign (セクション記号)	A78
Diaeresis (分音符号)	A8
Copyright sign (著作権記号)	A9
Feminine ordinal indicator (女性を示す標識)	AA
Left-pointing double quotation mark (左の二重引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Registered sign (登録商標)	AE
Macron (長音符号)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Acute accent (揚音アクセント)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Cedilla (セディーユ)	B8
Superscript one (上付きの 1)	B9
Masculine ordinal indicator (男性を示す標識)	BA
Right pointing double angle quotation mark (右の二重角引用符)	BB
Vulgar fraction one quarter (分数 4 分の 1)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Vulgar fraction three quarters (分数 4 分の 3)	BE
Inverted question mark (反転疑問符)	BF
Latin capital letter A with grave (抑音付きラテン大文字 A)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2

表 *24. ISO8859–9* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin capital letter A with tilde (波形記号付きラテン大文字 A)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter AE (ラテン大文字 AE)	C6
Latin capital letter C with cedilla (セディーユ付きラテン大文字 C)	C7
Latin capital letter E with grave (抑音付きラテン大文字 E)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter E with circumflex (曲折アクセント記号付きラテン大文字 E)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB
Latin capital letter I with grave (抑音付きラテン大文字 I)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter I with diaeresis (分音符号付きラテン大文字 I)	CF
Latin capital letter G with breve (短音記号付きラテン大文字 G)	D0
Latin capital letter N with tilde (波形記号付きラテン大文字 N)	D1
Latin capital letter O with grave (抑音付きラテン大文字 O)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with tilde (波形記号付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7
Latin capital letter O with stroke (斜線付きラテン大文字 O)	D8
Latin capital letter U with grave (抑音付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with circumflex (曲折アクセント記号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter I with dot above (上部ドット付きラテン大文字 I)	DD
Latin capital letter S with cedilla (セディーユ付きラテン大文字 S)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter a with grave (抑音付きラテン小文字 a)	E0
Latin small letter a with acute (揚音付きラテン小文字 a)	E1

表 *24. ISO8859–9* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with tilde (波形符号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter ae (ラテン小文字 ae)	E6
Latin small letter c with cedilla (セディーユ付きラテン小文字 c)	E7
Latin small letter e with grave (抑音付きラテン小文字 e)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with circumflex (曲折アクセント記号付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB
Latin small letter i with grave (抑音付きラテン小文字 i)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter i with diaeresis (分音符号付きラテン小文字 i)	EF
Latin small letter g with breve (短音記号付きラテン小文字 g)	F0
Latin small letter n with tilde (波形符号付きラテン小文字 n)	F1
Latin small letter o with grave (抑音付きラテン小文字 o)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with tilde (波形符号付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter o with stroke (斜線付きラテン小文字 o)	F8
Latin small letter u with grave (抑音付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with circumflex (曲折アクセント記号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter dotless i (ラテン小文字 ドットなし i)	FD
Latin small letter s with cedilla (セディーユ付きラテン小文字 s)	FE
Latin small letter y with diaeresis (分音符号付きラテン小文字 y)	FF

ISO8859–15

以下の情報では、ISO8859–15 用のコード・セットについて記述しています。

表 *25. ISO8859–15*

シンボル名	**16** 進値
No-break space (改行されないスペース)	A0
Inverted exclamation mark (反転感嘆符)	A1
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Euro sign (ユーロ記号)	A4
Yen sign (円記号)	A5
Latin capital letter S with caron (caron 付きラテン大文字 S)	A6
Section sign (セクション記号)	A7
Latin small letter s with caron (caron 付きラテン小文字 s)	A8
Copyright sign (著作権記号)	A9
Feminine ordinal indicator (女性を示す標識)	AA
Left-pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Registered sign (登録商標)	AE
Macron (長音符号)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Latin capital letter Z with caron (caron 付きラテン大文字 Z)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Latin small letter z with caron (caron 付きラテン小文字 z)	B8
Superscript one (上付きの 1)	B9
Masculine ordinal indicator (男性を示す標識)	BA
Right-pointing double angle quotation mark (右の二重角引用符)	BB
Latin capital ligature OE (ラテン大文字の合字 OE)	BC
Latin small ligature oe (ラテン小文字の合字 oe)	BD
Latin capital letter Y with diaeresis (分音符号付きラテン大文字 Y)	BE
Inverted question mark (反転疑問符)	BF
Latin capital letter A with grave (抑音付きラテン大文字 A)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2
Latin capital letter A with tilde (波形記号付きラテン大文字 A)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter AE (ラテン大文字 AE)	C6
Latin capital letter C with cedilla (セディーユ付きラテン大文字 C)	C7

表 25. ISO8859–15 (続き)

シンボル名	16 進値
Latin capital letter E with grave (抑音付きラテン大文字 E)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter W with circumflex (曲折アクセント記号付きラテン大文字 W)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB
Latin capital letter I with grave (抑音付きラテン大文字 I)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter I with diaeresis (分音符号付きラテン大文字 I)	CF
Latin capital letter eth (ラテン大文字 eth)	D0
Latin capital letter N with tilde (波形記号付きラテン大文字 N)	D1
Latin capital letter O with grave (抑音付きラテン大文字 O)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with tilde (波形記号付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7
Latin capital letter O with stroke (斜線付きラテン大文字 O)	D8
Latin capital letter U with grave (抑音付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with circumflex (曲折アクセント記号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter Y with acute (揚音付きラテン大文字 Y)	DD
Latin capital letter thorn (ラテン大文字 thorn)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter a with grave (抑音付きラテン小文字 a)	EO
Latin small letter a with acute (揚音付きラテン小文字 a)	E1
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with tilde (波形符号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter ae (ラテン小文字 ae)	E6
Latin small letter c with cedilla (セディーユ付きラテン小文字 c)	E7
Latin small letter e with grave (抑音付きラテン小文字 e)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with circumflex (曲折アクセント記号付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB
Latin small letter i with grave (抑音付きラテン小文字 i)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED

表 25. ISO8859–15 (続き)

シンボル名	16 進値
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter i with diaeresis (分音符号付きラテン小文字 i)	EF
Latin small letter eth (ラテン小文字 eth)	F0
Latin small letter n with tilde (波形符号付きラテン小文字 n)	F1
Latin small letter o with grave (抑音付きラテン小文字 o)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with tilde (波形符号付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter o with stroke (斜線付きラテン小文字 o)	F8
Latin small letter u with grave (抑音付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with circumflex (曲折アクセント記号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter y with acute (揚音付きラテン小文字 y)	FD
Latin small letter thorn (ラテン小文字 thorn)	FE
Latin small letter y with diaeresis (分音符号付きラテン小文字 y)	FF

IBM PC コード・セット

このセクションでは、IBM PC コード・セットについて説明します。

IBM-856

以下の情報では、IBM-856 用のコード・セットについて記述しています。

表 26. IBM-856 コード・セット

シンボル名	16 進値
Hebrew letter alef (ヘブライ文字 alef)	80
Hebrew letter bet (ヘブライ文字 bet)	81
Hebrew letter gimel (ヘブライ文字 gimel)	82
Hebrew letter dalet (ヘブライ文字 dalet)	83
Hebrew letter he (ヘブライ文字 he)	84
Hebrew letter vav (ヘブライ文字 vav)	85
Hebrew letter zayin (ヘブライ文字 zayin)	86
Hebrew letter het (ヘブライ文字 het)	87
Hebrew letter tet (ヘブライ文字 tet)	88
Hebrew letter yod (ヘブライ文字 yod)	89
Hebrew letter final kaf (ヘブライ文字 語尾の kaf)	8A
Hebrew letter kaf (ヘブライ文字 kaf)	8B
Hebrew letter lamed (ヘブライ文字 lamed)	8C
Hebrew letter final mem (ヘブライ文字 語尾の mem)	8D
Hebrew letter mem (ヘブライ文字 mem)	8E

表 *26. IBM-856* コード・セット *(*続き*)*

シンボル名	**16** 進値
Hebrew letter final nun (ヘブライ文字 語尾の nun)	8F
Hebrew letter nun (ヘブライ文字 nun)	90
Hebrew letter samekh (ヘブライ文字 samekh)	91
Hebrew letter ayin (ヘブライ文字 ayin)	92
Hebrew letter final pe (ヘブライ文字 語尾の pe)	93
Hebrew letter pe (ヘブライ文字 pe)	94
Hebrew letter final tsadi (ヘブライ文字 語尾の tsadi)	95
Hebrew letter tsadi (ヘブライ文字 tsadi)	96
Hebrew letter qof (ヘブライ文字 qof)	97
Hebrew letter resh (ヘブライ文字 resh)	98
Hebrew letter shin (ヘブライ文字 shin)	99
Hebrew letter tav (ヘブライ文字 tav)	9A
Pound sign (ポンド記号)	9C
Multiplication sign (乗算記号)	9E
Registered sign (登録商標)	A9
Not sign (否定記号)	AA
Vulgar fraction one half (分数 2 分の 1)	AB
Vulgar fraction one quarter (分数 4 分の 1)	AC
Left pointing double angle quotation mark (左の二重角引用符)	AE
Right pointing double angle quotation mark (右の二重角引用符)	AF
Light shade (薄い陰影)	B0
Medium shade (中間の陰影)	B1
Dark shade (暗い陰影)	B2
Box drawings light vertical (罫線記号 縦線)	B3
Box drawings light vertical and left (罫線記号 右端中央)	B4
Copyright sign (著作権記号)	B8
Box drawings double vertical and left (罫線記号 二重右端中央)	B9
Box drawings double vertical (罫線記号 二重縦線)	BA
Box drawings double down and left (罫線記号 二重上方右隅ボックス)	BB
Box drawings double up and left (罫線記号 二重下方右隅ボックス)	BC
Cent sign (セント記号)	BD
Yen sign (円記号)	BE
Box drawings light down and left (罫線記号 上方右隅ボックス)	BF
Box drawings light up and right (罫線記号 下方左隅ボックス)	C0
Box drawings light up and horizontal (罫線記号 下端中央)	C1
Box drawings light down and horizontal (罫線記号 上端中央)	C2
Box drawings light vertical and right (罫線記号 左端中央)	C3
Box drawings light horizontal (罫線記号 中央ボックス)	C4

表 *26. IBM-856* コード・セット *(*続き*)*

シンボル名	**16** 進値
Box drawings light vertical and horizontal (罫線記号 交差ボックス)	C5
Box drawings double up and right (罫線記号 二重下端左隅ボックス)	C8
Box drawings double down and right (罫線記号 二重上方左隅ボックス)	C9
Box drawings double up and horizontal (罫線記号 二重下端中央)	CA
Box drawings double down and horizontal (罫線記号 二重上端中央)	CB
Box drawings double vertical and right (罫線記号 二重左端中央)	CC
Box drawings double horizontal (罫線記号 二重中央ボックス)	CD
Box drawings double vertical and horizontal (罫線記号 二重交差ボックス)	CE
Currency sign (通貨記号)	CF
Box drawings light up and left (罫線記号 下方右隅ボックス)	D9
Box drawings light down and right (罫線記号 上方左隅ボックス)	DA
Full block (黒塗りボックス)	DB
Lower half block (下半分の黒塗りボックス)	DC
Broken bar (破線)	DD
Upper half block (上半分の黒塗りボックス)	DF
Micro sign (マイクロ記号)	E6
Overline (上線)	EE
Acute accent (揚音アクセント)	EF
Soft hyphen (ソフト・ハイフン)	F0
Plus-minus sign (プラス - マイナス記号)	F1
Double low line (二重下線)	F2
Vulgar fraction three quarters (分数 4 分の 3)	F3
Pilcrow sign (段落記号)	F4
Section sign (セクション記号)	F5
Division sign (除算記号)	F6
Cedilla (セディーユ)	F7
Degree sign (度記号)	F8
Diaeresis (分音符号)	F9
Middle dot (中黒)	FA
Superscript one (上付きの 1)	FB
Superscript three (上付きの 3)	FC
Superscript two (上付きの 2)	FD
Black square (黒塗りの正方形)	FE
No-break space (改行されないスペース)	FF

IBM-921

以下の情報では、IBM-921 用のコード・セットについて記述しています。

表 27. *IBM-921* コード・セット

シンボル名	**16** 進値
No-break space (改行されないスペース)	A0
Right double quotation mark (右の二重引用符)	A1
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Euro sign (ユーロ記号)	A4
Double low-9 quotation mark (一般記号 下付きの二重引用符)	A5
Broken bar (破線)	A6
Section sign (セクション記号)	A7
Latin capital letter O with stroke (斜線付きラテン大文字 O)	A8
Copyright sign (著作権記号)	A9
Latin capital letter R with cedilla (セディーユ付きラテン大文字 R)	AA
Left-pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Registered sign (登録商標)	AE
Latin capital letter AE (ラテン大文字 AE)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Left double quotation mark (左の二重引用符)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Latin small letter o with stroke (斜線付きラテン小文字 o)	B8
Superscript one (上付きの 1)	B9
Latin small letter r with cedilla (セディーユ付きラテン小文字 r)	BA
Right-pointing double angle quotation mark (右の二重角引用符)	BB
Vulgar fraction one quarter (分数 4 分の 1)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Vulgar fraction three quarters (分数 4 分の 3)	BE
Latin small letter ae (ラテン小文字 ae)	BF
Latin capital letter A with ogonek (ogonek 付きラテン大文字 A)	C0
Latin capital letter I with ogonek (ogonek 付きラテン大文字 I)	C1
Latin capital letter A with macron (長音符号付きラテン大文字 A)	C2
Latin capital letter C with acute (揚音付きラテン大文字 C)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4

表 27. *IBM-921* コード・セット (続き)

シンボル名	**16** 進値
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter E with ogonek (ogonek 付きラテン大文字 E)	C6
Latin capital letter E with macron (長音符号付きラテン大文字 E)	C7
Latin capital letter C with caron (caron 付きラテン大文字 C)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter Z with acute (揚音付きラテン大文字 Z)	CA
Latin capital letter E with dot above (上部ドット付きラテン大文字 E)	CB
Latin capital letter G with cedilla (セディーユ付きラテン大文字 G)	CC
Latin capital letter K with cedilla (セディーユ付きラテン大文字 K)	CD
Latin capital letter I with macron (長音符号付きラテン大文字 I)	CE
Latin capital letter L with cedilla (セディーユ付きラテン大文字 L)	CF
Latin capital letter S with caron (caron 付きラテン大文字 S)	D0
Latin capital letter N with acute (揚音付きラテン大文字 N)	D1
Latin capital letter N with cedilla (セディーユ付きラテン大文字 N)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with macron (長音符号付きラテン大文字 O)	D4
Latin capital letter O with tilde (波形記号付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7
Latin capital letter U with ogonek (ogonek 付きラテン大文字 U)	D8
Latin capital letter L with stroke (斜線付きラテン大文字 L)	D9
Latin capital letter S with acute (揚音付きラテン大文字 S)	DA
Latin capital letter U with macron (長音符号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter Z with dot above (上部ドット付きラテン大文字 Z)	DD
Latin capital letter Z with caron (caron 付きラテン大文字 Z)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter a with ogonek (ogonek 付きラテン小文字 a)	E0
Latin small letter i with ogonek (ogonek 付きラテン小文字 i)	E1

表 27. *IBM-921* コード・セット (続き)

シンボル名	**16** 進値
Latin small letter a with macron (長音符号付きラテン小文字 a)	E2
Latin small letter c with acute (揚音付きラテン小文字 c)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter e with ogonek (ogonek 付きラテン小文字 e)	E6
Latin small letter e with macron (長音符号付きラテン小文字 e)	E7
Latin small letter c with caron (caron 付きラテン小文字 c)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter z with acute (揚音付きラテン小文字 z)	EA
Latin small letter e with dot above (上部ドット付きラテン小文字 e)	EB
Latin small letter g with cedilla (セディーユ付きラテン小文字 g)	EC
Latin small letter k with cedilla (セディーユ付きラテン小文字 k)	ED
Latin small letter i with macron (長音符号付きラテン小文字 i)	EE
Latin small letter l with cedilla (セディーユ付きラテン小文字 l)	EF
Latin small letter s with caron (caron 付きラテン小文字 s)	F0
Latin small letter n with acute (揚音付きラテン小文字 n)	F1
Latin small letter n with cedilla (セディーユ付きラテン小文字 n)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with macron (長音符号付きラテン小文字 o)	F4
Latin small letter o with tilde (波形符号付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter u with ogonek (ogonek 付きラテン小文字 u)	F8
Latin small letter l with stroke (斜線付きラテン小文字 l)	F9
Latin small letter s with acute (揚音付きラテン小文字 s)	FA
Latin small letter u with macron (長音符号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter z with dot above (上部ドット付きラテン小文字 z)	FD
Latin small letter z with caron (caron 付きラテン小文字 z)	FE
Right single quotation mark (右の引用符)	FF

IBM-922

以下の情報では、IBM-922 用のコード・セットについて記述しています。

表 *28. IBM-922* コード・セット

シンボル名	**16** 進値
No break space (改行されないスペース)	A0
Inverted exclamation mark (反転感嘆符)	A1
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Euro sign (ユーロ記号)	A4
Yen sign (円記号)	A5
Broken bar (破線)	A6
Section sign (セクション記号)	A7
Diaeresis (分音符号)	A8
Copyright sign (著作権記号)	A9
Feminine ordinal indicator (女性を示す標識)	AA
Left-pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Registered sign (登録商標)	AE
Macron (長音符号)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Acute accent (揚音アクセント)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Cedilla (セディーユ)	B8
Superscript one (上付きの 1)	B9
Masculine ordinal indicator (男性を示す標識)	BA
Right-pointing double angle quotation mark (右の二重角引用符)	BB
Vulgar fraction one quarter (分数 4 分の 1)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Vulgar fraction three quarters (分数 4 分の 3)	BE
Inverted question mark (反転疑問符)	BF
Latin capital letter A with grave (抑音付きラテン大文字 A)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2
Latin capital letter A with tilde (波形記号付きラテン大文字 A)	C3
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter AE (ラテン大文字 AE)	C6

表 *28. IBM-922* コード・セット (続き)

シンボル名	**16** 進値
Latin capital letter C with cedilla (セディーユ付きラテン大文字 C)	C7
Latin capital letter E with grave (抑音付きラテン大文字 E)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter E with circumflex (曲折アクセント記号付きラテン大文字 E)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB
Latin capital letter I with grave (抑音付きラテン大文字 I)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter I with diaeresis (分音符号付きラテン大文字 I)	CF
Latin capital letter S with caron (caron 付きラテン大文字 S)	D0
Latin capital letter N with tilde (波形記号付きラテン大文字 N)	D1
Latin capital letter O with grave (抑音付きラテン大文字 O)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with tilde (波形記号付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7
Latin capital letter O with stroke (斜線付きラテン大文字 O)	D8
Latin capital letter U with grave (抑音付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with circumflex (曲折アクセント記号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter Y with acute (揚音付きラテン大文字 Y)	DD
Latin capital letter Z with caron (caron 付きラテン大文字 Z)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF
Latin small letter a with grave (抑音付きラテン小文字 a)	E0
Latin small letter a with acute (揚音付きラテン小文字 a)	E1
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with tilde (波形符号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter ae (ラテン小文字 ae)	E6
Latin small letter c with cedilla (セディーユ付きラテン小文字 c)	E7
Latin small letter e with grave (抑音付きラテン小文字 e)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with circumflex (曲折アクセント記号付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB

表 28. IBM-922 コード・セット (続き)

シンボル名	16 進値
Latin small letter i with grave (抑音付きラテン小文字 i)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter i with diaeresis (分音符号付きラテン小文字 i)	EF
Latin small letter s with caron (caron 付きラテン小文字 s)	F0
Latin small letter n with tilde (波形符号付きラテン小文字 n)	F1
Latin small letter o with grave (抑音付きラテン小文字 o)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with tilde (波形符号付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter o with stroke (斜線付きラテン小文字 o)	F8
Latin small letter u with grave (抑音付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with circumflex (曲折アクセント記号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter y with acute (揚音付きラテン小文字 y)	FD
Latin small letter z with caron (caron 付きラテン小文字 z)	FE
Latin small letter y with diaeresis (分音符号付きラテン小文字 y)	FF

IBM-1046

以下の情報では、IBM-1046 用のコード・セットについて記述しています。

表 29. IBM-1046

シンボル名	16 進値
Arabic letter alef with hamza below final form (アラビア文字 alef 下部 hamza 付き語尾形)	80
Multiplication sign (乗算記号)	81
Division sign (除算記号)	82
Arabic letter seen first part of final form (アラビア文字 seen 語尾形の最初の部分)	83
Arabic letter sheen first part of final form (アラビア文字 sheen 語尾形の最初の部分)	84
Arabic letter sad first part of final form (アラビア文字 sad 語尾形の最初の部分)	85
Arabic letter dad first part of final form (アラビア文字 dad 語尾形の最初の部分)	86
Arabic tatweel with fathatan above (アラビア文字 tatweel 上部 fathatan 付き)	87
Full block (黒塗りボックス)	89
Box drawings light vertical (罫線記号 縦線)	8A
Box drawings light horizontal (罫線記号 中央ボックス)	8B
Box drawings light down and left (罫線記号 上方右隅ボックス)	8C
Box drawings light down and right (罫線記号 上方左隅ボックス)	8D
Box drawings light up and right (罫線記号 下方左隅ボックス)	8E
Box drawings light up and left (罫線記号 下方右隅ボックス)	8F
Arabic damma medial form (アラビア文字 damma 語中形)	90

表 29. IBM-1046 (続き)

シンボル名	16 進値
Arabic kasra medial form (アラビア文字 kasra 語中形)	91
Arabic shadda medial form (アラビア文字 shadda 語中形)	92
Arabic sukun medial form (アラビア文字 sukun 語中形)	93
Arabic fatha medial form (アラビア文字 fatha 語中形)	94
Arabic letter yeh with hamza above final form (アラビア文字 yeh 上部 hamza 付き語尾形)	95
Arabic letter alef maksura final form (アラビア文字 alef maksura 語尾形)	96
Arabic letter yeh initial form (アラビア文字 yeh 語頭形)	97
Arabic letter yeh final form (アラビア文字 yeh 語尾形)	98
Arabic letter ghain final form (アラビア文字 ghain 語尾形)	99
Arabic letter ghain initial form (アラビア文字 ghain 語頭形)	9A
Arabic letter ghain medial form (アラビア文字 ghain 語中形)	9B
Arabic ligature lam with alef with madda above final form (アラビア文字の合字 lam 上部 madda 付き語尾形)	9C
Arabic ligature lam with alef with hamza above final form (アラビア文字 alef との合字 lam 上部 hamza 付き語尾形)	9D
Arabic ligature lam with alef with hamza below final form (アラビア文字 alef との合字 lam 下部 hamza 付き語尾形)	9E
Arabic ligature lam with alef final form (アラビア文字 alef との合字 lam 語尾形)	9f
No-break space (改行されないスペース)	A0
Arabic letter alef with madda above after lam (アラビア文字 lam の後に上部 madda 付き alef)	A1
Arabic letter alef with hamza above after lam (アラビア文字 lam の後に上部 hamza 付き alef)	A2
Arabic letter alef with hamza below after lam (アラビア文字 lam の後に下部 hamza 付き alef)	A3
Currency sign (通貨記号)	A4
Arabic letter alef after lam (アラビア文字 lam の後の alef)	A5
Arabic letter yeh with hamza above initial form (アラビア文字 yeh 上部 hamza 付き語頭形)	A6
Arabic letter beh with initial form (アラビア文字 beh 語頭形)	A7
Arabic letter teh with initial form (アラビア文字 teh 語頭形)	A8
Arabic letter theh with initial form (アラビア文字 theh 語頭形)	A9
Arabic letter jeem with initial form (アラビア文字 jeem 語頭形)	AA
Arabic letter hah with initial form (アラビア文字 hah 語頭形)	AB
Arabic comma (アラビア文字 コンマ)	AC
Soft hyphen (ソフト・ハイフン)	AD
Arabic letter khan initial form (アラビア文字 khan 語頭形)	AE
Arabic letter seen initial form (アラビア文字 seen 語頭形)	AF
Arabic-indic digit zero (アラビア・インド数字のゼロ)	B0
Arabic-indic digit one (アラビア・インド数字の 1)	B1
Arabic-indic digit two (アラビア・インド数字の 2)	B2
Arabic-indic digit three (アラビア・インド数字の 3)	B3
Arabic-indic digit four (アラビア・インド数字の 4)	B4
Arabic-indic digit five (アラビア・インド数字の 5)	B5
Arabic-indic digit six (アラビア・インド数字の 6)	B6
Arabic-indic digit seven (アラビア・インド数字の 7)	B7
Arabic-indic digit eight (アラビア・インド数字の 8)	B8
Arabic-indic digit nine (アラビア・インド数字の 9)	B9
Arabic letter sheen initial form (アラビア文字 sheen 語頭形)	BA

表 29. IBM-1046 (続き)

シンボル名	16 進値
Arabic semicolon (アラビア文字 セミコロン)	BB
Arabic letter sad initial form (アラビア文字 sad 語頭形)	BC
Arabic letter dad initial form (アラビア文字 dad 語頭形)	BD
Arabic letter ain initial form (アラビア文字 ain 語頭形)	BE
Arabic question mark (アラビア文字 疑問符)	BF
Arabic letter ain initial form (アラビア文字 ain 語頭形)	C0
Arabic letter hamza (アラビア文字 hamza)	C1
Arabic letter alef with madda above (上部 madda 付きアラビア文字 alef)	C2
Arabic letter alef with hamza above (上部 hamza 付きアラビア文字 alef)	C3
Arabic letter waw with hamza above (上部 hamza 付きアラビア文字 waw)	C4
Arabic letter alef with hamza below (下部 hamza 付きアラビア文字 alef)	C5
Arabic letter yeh with hamza above (上部 hamza 付きアラビア文字 yeh)	C6
Arabic letter alef (アラビア文字 alef)	C7
Arabic letter beh (アラビア文字 beh)	C8
Arabic letter teh marbuta (アラビア文字 teh marbuta)	C9
Arabic letter teh (アラビア文字 teh)	CA
Arabic letter theh (アラビア文字 theh)	CB
Arabic letter jeem (アラビア文字 jeem)	CC
Arabic letter hah (アラビア文字 hah)	CD
Arabic letter khah (アラビア文字 khah)	CE
Arabic letter dal (アラビア文字 dal)	CF
Arabic letter thal (アラビア文字 thal)	D0
Arabic letter reh (アラビア文字 reh)	D1
Arabic letter zain (アラビア文字 zain)	D2
Arabic letter seen (アラビア文字 seen)	D3
Arabic letter sheen (アラビア文字 sheen)	D4
Arabic letter sad (アラビア文字 sad)	D5
Arabic letter dad (アラビア文字 dad)	D6
Arabic letter tah (アラビア文字 tah)	D7
Arabic letter zah (アラビア文字 zah)	D8
Arabic letter ain (アラビア文字 ain)	D9
Arabic letter ghain (アラビア文字 ghain)	DA
Arabic letter ain medial form (アラビア文字 ain 語中形)	DB
Arabic letter alef with madda above final form (アラビア文字 alef 上部 madda 付き語尾形)	DC
Arabic letter alef with hamza above final form (アラビア文字 alef 上部 hamza 付き語尾形)	DD
Arabic letter alef with final form (アラビア文字 alef 語尾形付き)	DE
Arabic letter feh initial form (アラビア文字 feh 語頭形)	DF
Arabic tatweel (アラビア文字 tatweel)	E0
Arabic letter feh (アラビア文字 feh)	E1
Arabic letter qaf (アラビア文字 qaf)	E2
Arabic letter kaf (アラビア文字 kaf)	E3
Arabic letter lam (アラビア文字 lam)	E4
Arabic letter meem (アラビア文字 meem)	E5
Arabic letter noon (アラビア文字 noon)	E6
Arabic letter heh (アラビア文字 heh)	E7

表 29. IBM-1046 (続き)

シンボル名	16 進値
Arabic letter waw (アラビア文字 waw)	E8
Arabic letter alef maksura (アラビア文字 alef maksura)	E9
Arabic letter yeh (アラビア文字 yeh)	EA
Arabic fathatan (アラビア文字 fathatan)	EB
Arabic dammatan (アラビア文字 dammatan)	EC
Arabic kasratan (アラビア文字 kasratan)	ED
Arabic fatha (アラビア文字 fatha)	EE
Arabic damma (アラビア文字 damma)	EF
Arabic kasra (アラビア文字 kasra)	F0
Arabic shadda (アラビア文字 shadda)	F1
Arabic sukun (アラビア文字 sukun)	F2
Arabic letter qar initial form (アラビア文字 qar 語頭形)	F3
Arabic letter kaf initial form (アラビア文字 kaf 語頭形)	F4
Arabic letter lam initial form (アラビア文字 lam 語頭形)	F5
Arabic kasseh (アラビア文字 kasseh)	F6
Arabic ligature lam with alef with madda above isolated form (アラビア文字 alef との合字 lam 上部 madda 付き独立形)	F7
Arabic ligature lam with alef with hamza above isolated form (アラビア文字 alef との合字 lam 上部 hamza 付き独立形)	F8
Arabic ligature lam with alef with madda below isolated form (アラビア文字 alef との合字 lam 下部 madda 付き独立形)	F9
Arabic ligature lam with alef isolated form (アラビア文字 alef との合字 lam 独立形)	FA
Arabic letter meem initial form (アラビア文字 meem 語頭形)	FB
Arabic letter noon initial form (アラビア文字 noon 語頭形)	FC
Arabic letter heh initial form (アラビア文字 heh 語頭形)	FD
Arabic letter heh final form (アラビア文字 heh 語尾形)	FE
Euro sign (ユーロ記号)	FF

IBM-1124

以下の情報では、IBM-1124 用のコード・セットについて記述しています。

表 30. IBM-1124 コード・セット

シンボル名	16 進値
No-break space (改行されないスペース)	A0
Cyrillic capital letter io (キリル大文字 io)	A1
Cyrillic capital letter dje (キリル大文字 dje)	A2
Cyrillic capital letter ghe with upturn (上向き記号付きキリル大文字 ghe)	A3
Cyrillic capital letter ukrainian ie (キリル大文字 ウクライナ ie)	A4
Cyrillic capital letter dze (キリル大文字 dze)	A5
Cyrillic capital letter byelorussian-ukranian I (キリル大文字 ベラルーシ - ウクライナ I)	A6
Cyrillic capital letter yi (キリル大文字 yi)	A7
Cyrillic capital letter je (キリル大文字 je)	A8
Cyrillic capital letter lje (キリル大文字 lje)	A9

表 *30. IBM-1124* コード・セット *(*続き*)*

シンボル名	**16** 進値
Cyrillic capital letter nje (キリル大文字 nje)	AA
Cyrillic capital letter tshe (キリル大文字 tshe)	AB
Cyrillic capital letter kje (キリル大文字 kje)	AC
Soft hyphen (ソフト・ハイフン)	AD
Cyrillic capital letter short U (キリル大文字 ショート U)	AE
Cyrillic capital letter dzhe (キリル大文字 dzhe)	AF
Cyrillic capital letter A (キリル大文字 A)	B0
Cyrillic capital letter be (キリル大文字 be)	B1
Cyrillic capital letter ve (キリル大文字 ve)	B2
Cyrillic capital letter ghe (キリル大文字 ghe)	B3
Cyrillic capital letter de (キリル大文字 de)	B4
Cyrillic capital letter ie (キリル大文字 ie)	B5
Cyrillic capital letter zhe (キリル大文字 zhe)	B6
Cyrillic capital letter ze (キリル大文字 ze)	B7
Cyrillic capital letter I (キリル大文字 I)	B8
Cyrillic capital letter short I (キリル大文字 ショート I)	B9
Cyrillic capital letter ka (キリル大文字 ka)	BA
Cyrillic capital letter el (キリル大文字 el)	BB
Cyrillic capital letter em (キリル大文字 em)	BC
Cyrillic capital letter en (キリル大文字 en)	BD
Cyrillic capital letter O (キリル大文字 O)	BE
Cyrillic capital letter pe (キリル大文字 pe)	BF
Cyrillic capital letter er (キリル大文字 er)	C0
Cyrillic capital letter es (キリル大文字 es)	C1
Cyrillic capital letter te (キリル大文字 te)	C2
Cyrillic capital letter U (キリル大文字 U)	C3
Cyrillic capital letter ef (キリル大文字 ef)	C4
Cyrillic capital letter ha (キリル大文字 ha)	C5
Cyrillic capital letter tse (キリル大文字 tse)	C6
Cyrillic capital letter che (キリル大文字 che)	C7
Cyrillic capital letter sha (キリル大文字 sha)	C8
Cyrillic capital letter shcha (キリル大文字 shcha)	C9
Cyrillic capital letter hard sign (キリル大文字硬音記号)	CA
Cyrillic capital letter yeru (キリル大文字 yeru)	CB
Cyrillic capital letter soft sign (キリル大文字軟音記号)	CC
Cyrillic capital letter E (キリル大文字 E)	CD
Cyrillic capital letter yu (キリル大文字 yu)	CE
Cyrillic capital letter ya (キリル大文字 ya)	CF
Cyrillic small letter a (キリル小文字 a)	D0
Cyrillic small letter be (キリル小文字 be)	D1
Cyrillic small letter ve (キリル小文字 ve)	D2
Cyrillic small letter ghe (キリル小文字 ghe)	D3
Cyrillic small letter de (キリル小文字 de)	D4
Cyrillic small letter ie (キリル小文字 ie)	D5
Cyrillic small letter zhe (キリル小文字 zhe)	D6

表 *30. IBM-1124* コード・セット *(*続き*)*

シンボル名	**16** 進値
Cyrillic small letter ze (キリル小文字 ze)	D7
Cyrillic small letter i (キリル小文字 i)	D8
Cyrillic small letter short I (キリル小文字 ショート I)	D9
Cyrillic small letter ka (キリル小文字 ka)	DA
Cyrillic small letter el (キリル小文字 el)	DB
Cyrillic small letter em (キリル小文字 em)	DC
Cyrillic small letter en (キリル小文字 en)	DD
Cyrillic small letter o (キリル小文字 o)	DE
Cyrillic small letter pe (キリル小文字 pe)	DF
Cyrillic small letter er (キリル小文字 er)	E0
Cyrillic small letter es (キリル小文字 es)	E1
Cyrillic small letter te (キリル小文字 te)	E2
Cyrillic small letter u (キリル小文字 u)	E3
Cyrillic small letter ef (キリル小文字 ef)	E4
Cyrillic small letter ha (キリル小文字 ha)	E5
Cyrillic small letter tse (キリル小文字 tse)	E6
Cyrillic small letter che (キリル小文字 che)	E7
Cyrillic small letter sha (キリル小文字 sha)	E8
Cyrillic small letter shcha (キリル小文字 shcha)	E9
Cyrillic small letter hard sign (キリル小文字硬音記号)	EA
Cyrillic small letter yeru (キリル小文字 yeru)	EB
Cyrillic small letter soft sign (キリル小文字軟音記号)	EC
Cyrillic small letter e (キリル小文字 e)	ED
Cyrillic small letter yu (キリル小文字 yu)	EE
Cyrillic small letter ya (キリル小文字 ya)	EF
Numero sign (ナンバー記号)	F0
Cyrillic small letter io (キリル小文字 io)	F1
Cyrillic small letter dje (キリル小文字 dje)	F2
Cyrillic small letter ghe with upturn (上向き記号付きキリル小文字 ghe)	F3
Cyrillic small letter ukrainian ie (キリル小文字 ウクライナ ie)	F4
Cyrillic small letter dze (キリル小文字 dze)	F5
Cyrillic small letter byelorussian-ukrainian (キリル小文字 ベラルーシ - ウクライナ)	F6
Cyrillic small letter yi (キリル小文字 yi)	F7
Cyrillic small letter je (キリル小文字 je)	F8
Cyrillic small letter lje (キリル小文字 lje)	F9
Cyrillic small letter nje (キリル小文字 nje)	FA
Cyrillic small letter tshe (キリル小文字 tshe)	FB
Cyrillic small letter kje (キリル小文字 kje)	FC
Section sign (セクション記号)	FD
Cyrillic small letter short u (キリル小文字 ショート u)	FE
Cyrillic small letter dzhe (キリル小文字 dzhe)	FF

IBM-1129

以下の情報では、IBM-1129 用のコード・セットについて記述しています。

表 *31. IBM-1129* コード・セット

シンボル名	**16** 進値
No-break space (改行されないスペース)	A0
Inverted exclamation mark (反転感嘆符)	A1
Cent sign (セント記号)	A2
Pound sign (ポンド記号)	A3
Euro sign (ユーロ記号)	A4
Yen sign (円記号)	A5
Broken bar (破線)	A6
Section sign (セクション記号)	A7
Latin small ligature oe (ラテン小文字の合字 oe)	A8
Copyright sign (著作権記号)	A9
Feminine ordinal indicator (女性を示す標識)	AA
Left pointing double angle quotation mark (左の二重角引用符)	AB
Not sign (否定記号)	AC
Soft hyphen (ソフト・ハイフン)	AD
Registered sign (登録商標)	AE
Macron (長音符号)	AF
Degree sign (度記号)	B0
Plus-minus sign (プラス - マイナス記号)	B1
Superscript two (上付きの 2)	B2
Superscript three (上付きの 3)	B3
Latin capital letter Y with diaeresis (分音符号付きラテン大文字 Y)	B4
Micro sign (マイクロ記号)	B5
Pilcrow sign (段落記号)	B6
Middle dot (中黒)	B7
Latin capital ligature OE (ラテン大文字の合字 OE)	B8
Superscript one (上付きの 1)	B9
Masculine ordinal indicator (男性を示す標識)	BA
Right pointing double angle quotation mark (右の二重角引用符)	BB
Vulgar fraction one quarter (分数 4 分の 1)	BC
Vulgar fraction one half (分数 2 分の 1)	BD
Vulgar fraction three quarters (分数 4 分の 3)	BE
Inverted question mark (反転疑問符)	BF
Latin capital letter A with grave (抑音付きラテン大文字 A)	C0
Latin capital letter A with acute (揚音付きラテン大文字 A)	C1
Latin capital letter A with circumflex (曲折アクセント記号付きラテン大文字 A)	C2
Latin capital letter A with breve (短音記号付きラテン大文字 A)	C3

表 *31. IBM-1129* コード・セット *(*続き*)*

シンボル名	**16** 進値
Latin capital letter A with diaeresis (分音符号付きラテン大文字 A)	C4
Latin capital letter A with ring above (上部リング付きラテン大文字 A)	C5
Latin capital letter AE (ラテン大文字 AE)	C6
Latin capital letter C with cedilla (セディーユ付きラテン大文字 C)	C7
Latin capital letter E with grave (抑音付きラテン大文字 E)	C8
Latin capital letter E with acute (揚音付きラテン大文字 E)	C9
Latin capital letter E with circumflex (曲折アクセント記号付きラテン大文字 E)	CA
Latin capital letter E with diaeresis (分音符号付きラテン大文字 E)	CB
Combining grave accent (アクサングラーブ結合)	CC
Latin capital letter I with acute (揚音付きラテン大文字 I)	CD
Latin capital letter I with circumflex (曲折アクセント付きラテン大文字 I)	CE
Latin capital letter I with diaeresis (分音符号付きラテン大文字 I)	CF
Latin capital letter D with stroke (斜線付きラテン大文字 D)	D0
Latin capital letter N with tilde (波形記号付きラテン大文字 N)	D1
Combining hook above (上部フック結合)	D2
Latin capital letter O with acute (揚音付きラテン大文字 O)	D3
Latin capital letter O with circumflex (曲折アクセント記号付きラテン大文字 O)	D4
Latin capital letter O with horn (horn 付きラテン大文字 O)	D5
Latin capital letter O with diaeresis (分音符号付きラテン大文字 O)	D6
Multiplication sign (乗算記号)	D7
Latin capital letter O with stroke (斜線付きラテン大文字 O)	D8
Latin capital letter U with grave (抑音付きラテン大文字 U)	D9
Latin capital letter U with acute (揚音付きラテン大文字 U)	DA
Latin capital letter U with circumflex (曲折アクセント記号付きラテン大文字 U)	DB
Latin capital letter U with diaeresis (分音符号付きラテン大文字 U)	DC
Latin capital letter U with horn (horn 付きラテン大文字 U)	DD
Combining tilde (波形符号結合)	DE
Latin small letter sharp s (ラテン小文字 シャープ s)	DF

表 31. IBM-1129 コード・セット (続き)

シンボル名	16 進値
Latin small letter a with grave (抑音付きラテン小文字 a)	E0
Latin small letter a with acute (揚音付きラテン小文字 a)	E1
Latin small letter a with circumflex (曲折アクセント記号付きラテン小文字 a)	E2
Latin small letter a with breve (短音記号付きラテン小文字 a)	E3
Latin small letter a with diaeresis (分音符号付きラテン小文字 a)	E4
Latin small letter a with ring above (上部リング付きラテン小文字 a)	E5
Latin small letter ae (ラテン小文字 ae)	E6
Latin small letter c with cedilla (セディーユ付きラテン小文字 c)	E7
Latin small letter e with grave (抑音付きラテン小文字 e)	E8
Latin small letter e with acute (揚音付きラテン小文字 e)	E9
Latin small letter e with circumflex (曲折アクセント記号付きラテン小文字 e)	EA
Latin small letter e with diaeresis (分音符号付きラテン小文字 e)	EB
Combining acute accent (揚音アクセント結合)	EC
Latin small letter i with acute (揚音付きラテン小文字 i)	ED
Latin small letter i with circumflex (曲折アクセント記号付きラテン小文字 i)	EE
Latin small letter i with diaeresis (分音符号付きラテン小文字 i)	EF
Latin small letter d with stroke (斜線付きラテン小文字 d)	F0
Latin small letter n with tilde (波形符号付きラテン小文字 n)	F1
Combining dot below (下部ドット結合)	F2
Latin small letter o with acute (揚音付きラテン小文字 o)	F3
Latin small letter o with circumflex (曲折アクセント記号付きラテン小文字 o)	F4
Latin small letter o with horn (horn 付きラテン小文字 o)	F5
Latin small letter o with diaeresis (分音符号付きラテン小文字 o)	F6
Division sign (除算記号)	F7
Latin small letter o with stroke (斜線付きラテン小文字 o)	F8
Latin small letter u with grave (抑音付きラテン小文字 u)	F9
Latin small letter u with acute (揚音付きラテン小文字 u)	FA
Latin small letter u with circumflex (曲折アクセント記号付きラテン小文字 u)	FB
Latin small letter u with diaeresis (分音符号付きラテン小文字 u)	FC
Latin small letter u with horn (horn 付きラテン小文字 u)	FD
Dong sign (ドン記号)	FE
Latin small letter y with diaeresis (分音符号付きラテン小文字 y)	FF

TIS-620

以下の情報では、TIS-620 用のコード・セットについて記述しています。

表 *32. TIS-620* コード・セット

シンボル名	**16** 進値
Thai character ko kai (タイ文字 ko kai)	A1
Thai character kho khai (タイ文字 kho khai)	A2
Thai character kho khuat (タイ文字 kho khuat)	A3
Thai character kho khwai (タイ文字 kho khwai)	A4
Thai character kho khon (タイ文字 kho khon)	A5
Thai character kho rakhang (タイ文字 kho rakhang)	A6
Thai character ngo ngu (タイ文字 ngo ngu)	A7
Thai character cho chan (タイ文字 cho chan)	A8
Thai character cho ching (タイ文字 cho ching)	A9
Thai character cho chang (タイ文字 cho chang)	AA
Thai character so so (タイ文字 so so)	AB
Thai character cho choe (タイ文字 cho choe)	AC
Thai character yo ying (タイ文字 yo ying)	AD
Thai character do chada (タイ文字 do chada)	AE
Thai character to patak (タイ文字 to patak)	AF
Thai character tho than (タイ文字 tho than)	B0
Thai character tho nangmontho (タイ文字 tho nangmontho)	B1
Thai character tho phuthao (タイ文字 tho phuthao)	B2
Thai character no nen (タイ文字 no nen)	B3
Thai character do dek (タイ文字 do dek)	B4
Thai character to tao (タイ文字 to tao)	B5
Thai character tho thung (タイ文字 tho thung)	B6
Thai character tho thahan (タイ文字 tho thahan)	B7
Thai character tho thong (タイ文字 tho thong)	B8
Thai character no nu (タイ文字 no nu)	B9
Thai character bo baimai (タイ文字 bo baimai)	BA
Thai character po pla (タイ文字 po pla)	BB
Thai character pho phung (タイ文字 pho phung)	BC
Thai character fo fa (タイ文字 fo fa)	BD
Thai character pho phan (タイ文字 pho phan)	BE
Thai character fo fan (タイ文字 fo fan)	BF
Thai character pho samphao (タイ文字 pho samphao)	C0
Thai character mo ma (タイ文字 mo ma)	C1
Thai character yo yak (タイ文字 yo yak)	C2
Thai character ro rua (タイ文字 ro rua)	C3
Thai character ru (タイ文字 ru)	C4
Thai character lo ling (タイ文字 lo ling)	C5
Thai character lu (タイ文字 lu)	C6
Thai character wo waen (タイ文字 wo waen)	C7
Thai character so sala (タイ文字 so sala)	C8
Thai character so rusi (タイ文字 so rusi)	C9

表 32. *TIS-620* コード・セット (続き)

シンボル名	**16** 進値
Thai character so sua (タイ文字 so sua)	CA
Thai character ho hip (タイ文字 ho hip)	CB
Thai character lo chula (タイ文字 lo chula)	CC
Thai character o ang (タイ文字 o ang)	CD
Thai character ho nokhuk (タイ文字 ho nokhuk)	CE
Thai character paiyannoi (タイ文字 paiyannoi)	CF
Thai character sara a (タイ文字 sara a)	D0
Thai character mai han-akat (タイ文字 mai han-akat)	D1
Thai character sara aa (タイ文字 sara aa)	D2
Thai character sara am (タイ文字 sara am)	D3
Thai character sara i (タイ文字 sara i)	D4
Thai character sara ii (タイ文字 sara ii)	D5
Thai character sara ue (タイ文字 sara ue)	D6
Thai character sara uee (タイ文字 sara uee)	D7
Thai character sara u (タイ文字 sara u)	D8
Thai character uu (タイ文字 uu)	D9
Thai character phinthu (タイ文字 phinthu)	DA
Thai currency symbol baht (タイ通貨記号バーツ)	DF
Thai character sara e (タイ文字 sara e)	E0
Thai character sara ae (タイ文字 sara ae)	E1
Thai character sara O (タイ文字 sara O)	E2
Thai character sara ai maimuan (タイ文字 sara ai maimuan)	E3
Thai character sara ai maimalai (タイ文字 sara ai maimalai)	E4
Thai character lakkhangyao (タイ文字 lakkhangyao)	E5
Thai character maiyamok (タイ文字 maiyamok)	E6
Thai character maitaikhu (タイ文字 maitaikhu)	E7
Thai character mai ek (タイ文字 mai ek)	E8
Thai character mai tho (タイ文字 mai tho)	E9
Thai character mai tri (タイ文字 mai tri)	EA
Thai character mai chattawa (タイ文字 mai chattawa)	EB
Thai character thanthakhat (タイ文字 thanthakhat)	EC
Thai character nikhahit (タイ文字 nikhahit)	ED
Thai character yamakkan (タイ文字 yamakkan)	EE
Thai character fongman (タイ文字 fongman)	EF
Thai digit zero (タイ数字のゼロ)	F0
Thai digit one (タイ数字 1)	F1
Thai digit two (タイ数字 2)	F2
Thai digit three (タイ数字 3)	F3
Thai digit four (タイ数字 4)	F4
Thai digit five (タイ数字 5)	F5
Thai digit six (タイ数字 6)	F6
Thai digit seven (タイ数字 7)	F7
Thai digit eight (タイ数字 8)	F8

表 32. TIS-620 コード・セット (続き)

シンボル名	16 進値
Thai digit nine (タイ数字 9)	F9
Thai character angkhankhu (タイ文字 angkhankhu)	FA
Thai character khomut (タイ文字 khomut)	FB

各国語サポートのサンプル・プログラム

このセクションには、サンプル・プログラムの一部である my_example.c が含まれています。これは、コード・セットから独立したプログラミングによるグローバリゼーションを示しています。

関連概念:

12 ページの『コード・セットからの独立性』
システムが外部の環境と通信するには、コード・セットに関する特定の情報が必要です。 この情報は、コード・セットから独立したライブラリー・サブルーチン (グローバリゼーション・ライブラリー) によって隠されています。この種のサブルーチンは、コード・セットに従属する関数に情報を渡します。各国語サポート・サブルーチンが必要なコード・セット情報を処理するため、コード・セットに関する十分な知識がなくても、文字を処理するプログラムを作成できます。 このようなプログラミング手法のことを、コード・セットの独立性 といいます。

my_example 用のメッセージ・ソース・ファイル

ここでは、my_example ユーティリティーのサンプル・メッセージ・ソース・ファイルを示します。 このカタログには 1 つのセットと 3 つのメッセージのみを定義しましたが、これは単に例を示しているだけであることに注意してください。 代表的なカタログには、この種のメッセージがいくつか入っています。

次に、my_example 用のメッセージ・ソース・ファイル my_example.msg を示します。

```
$quote "
$set MS_MY_EXAMPLE
CANTOPEN        "my_example: cannot open %s¥n"
BYTECNT         "number of bytes: %d¥n"
CHARCNT         "number of characters: %d
```

my_example 用のメッセージ・ヘッダー・ファイルの作成

実行時カタログを生成するには、**runcat** コマンドを次のように使用します。

```
runcat my_example my_example.msg
```

これは、以下のセクションに示す **my_example_msg.h** ヘッダー・ファイルを生成します。 セット・ニーモニックは MS_MY_EXAMPLE、メッセージ・ニーモニックは CANTOPEN、 BYTECNT、および CHARCNT であることに注意してください。これらのニーモニックは、この付録のプログラムで使用されます。

```
/*
** The header file: my_example_msg.h is as follows:
*/

#ifndef _H_MY_EXAMPLE_MSG
#define _H_MY_EXAMPLE_MSG
#include <limits.h>
#include <nl_types.h>
#define MF_MY_EXAMPLE "my_example.cat"

/* The following was generated from my_example.msg. */
```

```
/* definitions for set MS_MY_EXAMPLE */
#define MS_MY_EXAMPLE 1

#define CANTOPEN 1
#define BYTECNT  2
#define CHARCNT  3

#endif
```

コード・セットから独立した単一ソース単一パスのバージョン

単一ソース単一パス *(single-source single-path)* という用語は、単一バイトとマルチバイト・コード・セットの両方の処理に使用する、単一アプリケーションの 1 つのパスを意味します。 単一ソース単一パス・メソッドを用いると、グローバリゼーションのためのすべての **ifdef** が除去されます。単一バイト・コード・セットのメンバーであるかマルチバイト・コード・セットのメンバーであるかに関係なく、すべての文字が同様に処理されます。

単一ソース単一パスは望ましいが、パフォーマンスは低下することがあります。したがって、この方式はすべてのプログラムにはお勧めできません。 完全に国際化されているプログラムであれば、パフォーマンス低下が生じない場合もあります。このような場合には、単一ソース単一パス・メソッドを使用してください。

次に示す完全な国際化バージョンの my_example ユーティリティーは、単一ソース単一パスの、コード・セット独立のプログラミングにより、すべてのコード・セットをサポートします。

```
/*
 * COMPONENT_NAME:
 *
 * FUNCTIONS: my_example
 *
 * The following code shows how to count the number of bytes and
 * the number of characters in a text file.
 *
 * This example is for illustration purposes only. Performance
 * improvements may still be possible.
 *
 */

#include        <stdio.h>
#include        <ctype.h>
#include        <locale.h>
#include        <stdlib.h>
#include        "my_example_msg.h"

#define MSGSTR(Num,Str) catgets(catd,MS_MY_EXAMPLE,Num,Str)

/*
 * NAME: my_example
 *
 * FUNCTION: Counts the number of characters in a file.
 *
 */

main(argc,argv)
int argc;
char **argv;
{
    int     bytesread,   /* number of bytes read */
        bytesprocessed;
    int     leftover;

   int     i;
    int     mbcnt;           /* number of bytes in a character */
```

```
 int     f;               /* File descriptor */
 int mb_cur_max;
 int    bytect;           /* name changed from charct... */
 int    charct;           /* for real character count */
 char   *curp, *cure;     /* current and end pointers into
                             ** buffer */
 char         buf[BUFSIZ+1];

 nl_catd      catd;

 wchar_t    wc;

 /* Obtain the current locale */
 (void) setlocale(LC_ALL,"");

 /* after setting the locale, open the message catalog */
 catd = catopen(MF_MY_EXAMPLE,NL_CAT_LOCALE);

 /* Parse the arguments if any */

 /*
 ** Obtain  the maximum number of bytes in a character in the
 ** current locale.
 */
 mb_cur_max = MB_CUR_MAX;
 i = 1;

 /* Open the specified file and issue error messages if any */
 f = open(argv[i],0);
 if(f<0){
     fprintf(stderr,MSGSTR(CANTOPEN,              /*MSG*/
         "my_example: cannot open %s¥n"), argv[i]);      /*MSG*/
         exit(2);
 }

/* Initialize the variables for the count */
 bytect = 0;
 charct = 0;

 /* Start count of bytes and characters  */

 leftover = 0;

 for(;;) {
     bytesread = read(f,buf+leftover, BUFSIZ-leftover);
     /* issue any error messages here, if needed */
     if(bytesread <= 0)
          break;

     buf[leftover+bytesread] = '¥0';
             /* Protect partial reads */
     bytect += bytesread;
     curp=buf;
     cure = buf + bytesread+leftover;
     leftover=0;      /* No more leftover */

     for(; curp<cure ;){
         /* Convert to wide character */
         mbcnt= mbtowc(&wc, curp, mb_cur_max);
         if(mbcnt <= 0){
             mbcnt = 1;
         }else if (cure - curp >=mb_cur_max){
             wc = *curp;
             mbcnt =1;
         }else{
             /* Needs more data */
             leftover= cure - curp;
```

```
                strcpy(buf, curp, leftover);
                break;
            }
            curp +=mbcnt;
            charct++;
        }
    }

        /* print number of chars and bytes */
    fprintf(stderr,MSGSTR(BYTECNT, "number of bytes:%d¥n"),
            bytect);
    fprintf(stderr,MSGSTR(CHARCNT, "number of characters:%d¥n"),
            charct);
    close(f);
    exit(0);
}
```

単一バイト・コード・セット用に最適化された単一ソース二重パス・バージョン

単一ソース二重パス *(single-source dual-path)* という用語は、単一アプリケーションにおける 2 つのパスを意味します。このアプリケーションでは、実行時に、使用中のコード・セットが単一バイトかマルチバイトかを示す現行ロケールの設定に応じて、いずれかのパスが選択されます。

プログラムがそのパフォーマンスを維持し、実行可能ファイルのサイズもあまり増やさないようにできれば、単一ソース二重パス・メソッドが望ましい選択といえます。 実行可能ファイルのサイズの増加を、コマンド単位かユーティリティー単位で評価する必要があります。

単一ソース二重パス・メソッドでは、**MB_CUR_MAX** マクロが、現行ロケールのマルチバイト文字の最大バイト数を指定します。これは、選択する処理パスが単一バイトかマルチバイトかを、実行時に判別するときに使用すべきです。 例えば次の例のように、ブール・フラグを用いて選択するパスを示します。

```
int mbcodeset ;
/* After setlocale(LC_ALL,"") is done, determine the path to
** be chosen.
*/
if(MB_CUR_MAX == 1)
        mbcodeset = 0;
else    mbcodeset = 1;
```

このように、現行コード・セットを検査して、これがマルチバイト・コード・セットかどうかを調べ、マルチバイト・コード・セットであれば、フラグ mbcodeset が適切に設定されます。パフォーマンスへの影響は、**MB_CUR_MAX** マクロを何回もテストするよりこのフラグのテストのほうが少なく済みます。

```
if(mbcodeset){
        /* Multibyte code sets (also supports single-byte
        ** code sets )
        */
        /* Use multibyte or wide character processing
        functions */
}else{
        /* single-byte code sets */
        /* Process accordingly */
}
```

グローバリゼーションがモジュールの小さい比率に影響を与える場合は、上記の方法が適切です。二重パスを備えるための過度のテストは、パフォーマンスを低下させる可能性があります。 この場合は、テストが多くならないレベルで、テストを行います。

my_example ユーティリティーの次に示すバージョンは、 1 つのオブジェクトを作成しますが、実行時には、コード・セットに基づいて適切なパスが選択され、そのコード・セットのパフォーマンスを最適化します。 単一バイト・コード・セットとマルチバイト・コード・セットの区別だけをしていることに注意してください。

```
/*
 * COMPONENT_NAME:
 *
 * FUNCTIONS: my_example
 *
 * The following code shows how to count the number of bytes and
 * the number of characters in a text file.
 *
 * This example is for illustration purposes only. Performance
 * improvements may still be possible.
 *
 */

#include        <stdio.h>
#include        <ctype.h>
#include        <locale.h>
#include        <stdlib.h>
#include        "my_example_msg.h"

#define MSGSTR(Num,Str) catgets(catd,MS_MY_EXAMPLE,Num,Str)

/*
 * NAME: my_example
 *
 * FUNCTION: Counts the number of characters in a file.
 *
 */

main(argc,argv)
int argc;
char **argv;
{
    int bytesread,  /* number of bytes read */
        bytesprocessed;
    int   leftover;

    int   i;
    int   mbcnt;   /* number of bytes in a character */
    int   f;       /* File descriptor */
    int   mb_cur_max;
    int    bytect;             /* name changed from charct... */
    int    charct;             /* for real character count */
    char   *curp, *cure; /* current and end pointers into buffer */
    char        buf[BUFSIZ+1];

        nl_catd          catd;

        wchar_t    wc;

        /* flag to indicate if current code set is a
        ** multibyte code set
        */
        int     multibytecodeset;

        /* Obtain the current locale */
        (void) setlocale(LC_ALL,"");

    /* after setting the locale, open the message catalog */
    catd = catopen(MF_MY_EXAMPLE,NL_CAT_LOCALE);

   /* Parse the arguments if any */
```

```
/*
** Obtain the maximum number of bytes in a character in the
** current locale.
*/
mb_cur_max = MB_CUR_MAX;

if(mb_cur_max >1)
    multibytecodeset = 1;
else
    multibytecodeset = 0;

i = 1;

/* Open the specified file and issue error messages if any */
f = open(argv[i],0);
if(f<0){
    fprintf(stderr,MSGSTR(CANTOPEN,              /*MSG*/
        "my_example: cannot open %s¥n"), argv[i]);      /*MSG*/
        exit(2);
}

/* Initialize the variables for the count */
bytect = 0;
charct = 0;

/* Start count of bytes and characters  */

leftover = 0;

if(multibytecodeset){
    /* Full globalization */
    /* Handles supported multibyte code sets */
    for(;;) {
        bytesread = read(f,buf+leftover,
                BUFSIZ-leftover);
        /* issue any error messages here, if needed */
        if(bytesread <= 0)
            break;

        buf[leftover+bytesread] = '¥0';
                /* Protect partial reads */
        bytect += bytesread;
        curp=buf;

        cure = buf + bytesread+leftover;
        leftover=0; /* No more leftover */

        for(; curp<cure ;){
            /* Convert to wide character */
            mbcnt= mbtowc(&wc, curp, mb_cur_max);
            if(mbcnt <= 0){
                mbcnt = 1;
            }else if (cure - curp >=mb_cur_max){
                wc = *curp;
                mbcnt =1;

            }else{
                /* Needs more data */
                leftover= cure - curp;
                strcpy(buf, curp, leftover);
                break;
            }
            curp +=mbcnt;
            charct++;
        }
    }
```

```
    }else {

        /* Code specific to single-byte code sets that
        ** avoids conversion to widechars and thus optimizes
        ** performance for single-byte code sets.
        */

        for(;;) {
            bytesread = read(f,buf, BUFSIZ);
            /* issue any error messages here, if needed */
            if(bytesread <= 0)
                    break;

                bytect += bytesread;
                charct += bytesread;
        }

    }

        /* print number of chars and bytes */
    fprintf(stderr,MSGSTR(BYTECNT, "number of bytes:%d¥n"),
            bytect);
    fprintf(stderr,MSGSTR(CHARCNT, "number of characters:%d¥n"),
            charct);
    close(f);
    exit(0);
}
```

libcur パッケージの使用

このセクションでは、libcur パッケージ (AT&T の libcurses パッケージの拡張機能) を使用するプログラムに対して行う必要のある変更について説明します。

libcur パッケージ (AT&T の libcurses パッケージの拡張機能) を使用するプログラムは、以下の変更を行う必要があります。

1. コード・セット内の文字を表すのに必要なバイト数は、文字の表示列幅も表すという前提を除去します。 文字のワイド文字コードに必要な表示列数は、 **wcwidth** サブルーチンを用いて判別します。
2. **NLSCHAR** を **wchar_t** として再定義します。
3. `win->_y [y][x]` に **wchar_t** エンコードがあります。
4. プログラムは、**wchar_t** に特定のエンコードを想定してはなりません。
5. プログラムは、**addch** サブルーチン・ファミリーではなく、**addstr**、**waddstr**、**mvaddstr**、および **mvwaddstr** サブルーチンを使用する必要があります。 ストリング引数は、すべてマルチバイト形式をとります。
6. **addch** および **waddch** サブルーチンは、文字の **wchar_t** エンコードを受け入れます。 これらのサブルーチンを使用するプログラムでは、これらの関数を呼び出す際には必ず **wchar_t** を使用する必要があります。 (x,y) は、これらのサブルーチンに渡される **wchar_t** が占有する列数だけ増やされます。
7. **delch**、**wdelch**、**mvdelch**、および **mvwdelch** サブルーチンは、現在位置 (x,y) に応じたマルチバイト文字の削除およびバックスペースをサポートします。 現在の (x,y) の列位置が、2 列文字の 1 番目か 2 番目の列を指し示す場合は、 **delch** サブルーチンにより、両方の列を削除して、削除された列数だけ行の残りの列をシフトします。
8. **insch**、**winsch**、**mvinsch**、および **mvwinsch** サブルーチンは、文字の wchar_t エンコードを現在位置 (x,y) に挿入する場合に使用することができます。 行は、wchar_t に必要な列数だけシフトされます。

9. libcur パッケージは **terminfo** データベースで定義されたボックス描画文字をサポートするように変更されており、 IBM-850 コード・セットのグラフィック文字は想定していません。 libcur パッケージは、terminfo データベースの **box_chars_1** および **box_chars_2** エントリーで定義された、 1 次ボックス文字および代替ボックス文字をサポートします。 これを使用するには、プログラムを次のように変更する必要があります。

   描画、1 次ボックス文字:

   ```
   wcolorout(win, Bxa);
   cbox(win);
   wcolorend(win);

   または
   wcolorout(win, Bxa);
   drawbox(win, y,x, height, width);
   wcolorend(win);
   ```

   描画、代替ボックス文字

   ```
   wcolorout(win, Bya)
   cboxalt(win);
   wcolorend(win);

   または

   wcolorout(win, Bya);
   drawbox(win, y, x, height, width);
   wcolorend(win);
   ```

   Bxa および Bya は、 **terminfo** データベースで定義された 1 次属性および代替属性を参照します。

   cur01.h ファイルに、次のマクロが追加されます。

   ```
   cboxalt(win)

   drawboxalt(win, y,x, height, width)
   ```

10. マルチバイト文字の入力をサポートする必要があるプログラムは、 **extended(TRUE)** のコールで **_extended** を TRUE に設定してはなりません。 **_extended** フラグが真のとき、 **wgetch** サブルーチンは、文字の **wchar_t** エンコードを戻します。 マルチバイト文字の場合、この **wchar_t** のエンコードは、ファンクション・キーのエスケープ・シーケンスの事前定義値と矛盾する場合があります。 マルチバイト・コード・セットを使用するときは、入力の前に拡張をオフに設定する (**extended(FALSE)**) ことによって、この矛盾を回避してください。 (デフォルトは TRUE です。)

    マルチバイト文字入力を行うプログラムは、次のようにする必要があります。

    ```
    Input routine:

    Example:

        int c, count;
        char buf[];

        extended(FALSE); /* obtain one byte at a time */
        count =0;
        while(1){
            c = wgetch();  /* get one byte at a time */
            buf[count++] = c;
            if(count <=MB_CUR_MAX)
                if(mblen(buf, count) != -1)
    ```

```
                    break; /* character found* /
            else
                /*Error. No character can be found */
                /* Handle this case appropriately */
                break;
        }
/* buf contains the input multibyte sequence */
/* Now handle PF keys, or any escape sequence here */
```

11. **inch**、**winch**、**mvinch**、および **mvwinch** サブルーチンは、現在位置 (x,y) で **wchar_t** を戻します。 2 列幅の文字に関しては、(x,y) ポイントが最初の列である場合は、2 列幅の文字の **wchar_t** コードが戻されますので、注意してください。 (x,y) ポイントが 2 番目の列である場合は WEOF が戻されます。

特記事項

本書は米国が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービス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、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービス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 は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述は、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年).

このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie などの各種テクノロジーの使用について詳しくは、『IBM オンラインでのプライバシー・ステートメントのハイライト』(http://www.ibm.com/privacy/jp/ja/)、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』というタイトルのセクション、および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Microsoft、および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

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。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

［サ行］

［タ行］

［ナ行］

［ハ行］

［マ行］

［ヤ行］

［ラ行］

［ワ行］

[数字]

A

B

C

E

F

G

I

K

L

M

N

P

R

S

T

W

［特殊文字］

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21