

# Tru64 UNIX

---

## 国際化ソフトウェア・プログラミング・ガイド

Part Number: AA-RK3SC-TE

**2002 年 11 月**

ソフトウェア・バージョン: Tru64 UNIX Version 5.1B 以上

本書では、国際化ソフトウェアの作成方法と Tru64 UNIX オペレーティング・システムで提供している国際化プログラム作成支援ツールについて説明します。

---

© 2002 日本ヒューレット・パッカート株式会社

本書の著作権は日本ヒューレット・パッカート株式会社が保有しており、本書中の解説および図、表は日本ヒューレット・パッカートの文書による許可なしに、その全体または一部を、いかなる場合にも再版あるいは複製することを禁じます。

また、本書に記載されている事項は、予告なく変更されることがありますので、あらかじめご承知おください。万一、本書の記述に誤りがあった場合でも、弊社は一切その責任を負いかねます。

日本ヒューレット・パッカートは、弊社または弊社の指定する会社から納入された機器以外の機器で対象ソフトウェアを使用した場合、その性能あるいは信頼性について一切責任を負いかねます。

本書で解説するソフトウェア(対象ソフトウェア)は、所定のライセンス契約が締結された場合に限り、その使用あるいは複製が許可されます。

Open Software Foundation, OSF, OSF/1, OSF/Motif, および Motif は Open Software Foundation 社の商標です。UNIX は The Open Group の米国ならびに他の国における登録商標です。The Open Group は The Open Group の米国ならびに他の国における商標です。X/Open は X/Open カンパニーリミテッドの商標です。Adobe, Acrobat Reader, PostScript, および Display PostScript は米国 Adobe Systems 社の登録商標です。

COMPAQ, Compaq ロゴ, Digital ロゴは U.S. Patent and Trademark Office に登録されています。以下は、Digital Equipment Corporation の商標です: ALL-IN-1, Alpha AXP, AlphaGeneration, AlphaServer, AltaVista, ATMworks, AXP, Bookreader, CDA, DDIS, DEC, DEC Ada, DECevent, DEC Fortran, DEC FUSE, DECnet, DECstation, DECsystem, DECterm, DECUS, DECwindows, DTIF, Massbus, MicroVAX, OpenVMS, POLYCENTER, PrintServer, Q-bus, StorageWorks, Tru64, TruCluster, TURBOchannel, ULTRIX, ULTRIX Mail Connection, ULTRIX Worksystem Software, UNIBUS, VAX, VAXstation, VMS, XUI。このドキュメントに記載されているその他の会社名および製品名は、各社の商標または登録商標です。

原典 Writing Software for the International Market (AA-RH9YC-TE)  
©2002 Hewlett-Packard Company

---

# 目次

## まえがき

## 1 国際化ソフトウェア開発の概要

1.1	言語宣言 .....	1-1
1.1.1	地域化 .....	1-2
1.2	言語 .....	1-3
1.2.1	文字分類 .....	1-3
1.2.2	大文字と小文字の変換 .....	1-4
1.2.3	メッセージ・カタログ .....	1-4
1.3	文化的データ .....	1-4
1.3.1	言語情報 .....	1-4
1.4	文字セット .....	1-5
1.4.1	照合順序 .....	1-5
1.4.2	文字と文字列 .....	1-6
1.4.3	ポータブル文字セット .....	1-7
1.4.4	ユニバーサル文字セット .....	1-8

## 2 国際化ソフトウェアの開発

2.1	ロケールの使用 .....	2-2
2.2	コードセットの使用 .....	2-4
2.2.1	データの透過性の確保 .....	2-11
2.2.2	コード内リテラルの使用 .....	2-11
2.2.3	マルチバイト文字の操作 .....	2-13
2.2.4	マルチバイト文字データとワイド文字データ間の変換 .....	2-13

2.2.5	ソースおよび実行コードセットにおけるマルチバイト文字の規則 .....	2-15
2.2.6	文字の分類 .....	2-16
2.2.7	文字の変換 .....	2-17
2.2.8	文字列の比較 .....	2-17
2.3	文化的データの処理 .....	2-19
2.3.1	langinfo データベース .....	2-20
2.3.2	langinfo データベースの照会 .....	2-20
2.3.3	ローカルな習慣に合った日付および時刻文字列の生成と解釈 .....	2-21
2.3.4	金額値の書式付け .....	2-22
2.3.5	プログラム独自の形式による数値の書式付け .....	2-22
2.3.6	他の処理における langinfo データベースの使用 .....	2-23
2.4	テキストの表示と入力の処理 .....	2-23
2.4.1	メッセージの作成と使用 .....	2-24
2.4.2	出力テキストの書式付け .....	2-26
2.4.3	入力テキストの走査 .....	2-27
2.5	実行時環境へのロケールのバインド .....	2-28
2.5.1	システムまたはユーザのロケール・セットへのバインド ..	2-29
2.5.2	プログラム実行時のロケールの変更 .....	2-30

### 3 メッセージ・カタログの作成と使用

3.1	メッセージ・テキスト・ソース・ファイルの作成 .....	3-2
3.1.1	一般規則 .....	3-5
3.1.2	メッセージ・セット .....	3-7
3.1.3	メッセージ・エントリ .....	3-10
3.1.4	quote ディレクティブ .....	3-12
3.1.5	コメント行 .....	3-13
3.1.6	メッセージ・スタイルのガイドライン .....	3-14

3.2	既存のプログラムからのメッセージ・テキストの抽出 .....	3-18
3.3	メッセージ・ソース・ファイルの編集と翻訳 .....	3-20
3.4	メッセージ・カタログの生成 .....	3-22
3.4.1	mkcatdefs コマンドの使用法 .....	3-23
3.4.2	gencat コマンドの使用法 .....	3-26
3.4.3	メッセージ・カタログの設計と保守に関する留意点 .....	3-27
3.5	メッセージとロケール・データの表示 .....	3-32
3.6	プログラムからのメッセージ・カタログへのアクセス .....	3-34
3.6.1	メッセージ・カタログのオープン .....	3-34
3.6.2	メッセージ・カタログのクローズ .....	3-40
3.6.3	プログラム・メッセージの読み取り .....	3-40
<b>4</b>	<b>curses ライブラリ・ルーチンを使用したワイド文字の処理</b>	
4.1	curses ウィンドウへのワイド文字の書き込み .....	4-2
4.1.1	ワイド文字を追加 (上書き) し, カーソルを進める .....	4-2
4.1.2	ワイド文字を (上書きせずに) 挿入し, カーソルを進めない	4-3
4.2	curses ウィンドウへのワイド文字列の書き込み .....	4-4
4.2.1	ワイド文字列を追加 (上書き) し, カーソルを進めない ...	4-4
4.2.2	ワイド文字列を追加 (上書き) し, カーソルを進める .....	4-5
4.2.3	ワイド文字列を (上書きせずに) 挿入し, カーソルを進めない .....	4-7
4.3	curses ウィンドウからのワイド文字の削除 .....	4-8
4.4	curses ウィンドウからのワイド文字の読み取り .....	4-9
4.5	curses ウィンドウからのワイド文字列の読み取り .....	4-10
4.5.1	属性付きのワイド文字列の読み取り .....	4-10
4.5.2	属性なしでのワイド文字列の読み取り .....	4-11
4.6	端末からの文字列の読み取り .....	4-12
4.7	キーボードからのワイド文字の読み取りまたはキューイング	4-14
4.8	curses ウィンドウでの書式付きテキストの変換 .....	4-15

4.9	curses ウィンドウでの書式付きテキストの表示 .....	4-16
<b>5</b>	<b>国際化対応の X , Xt , および Motif アプリケーションの作成</b>	
5.1	Xt インタリックス・ライブラリの国際化機能の使用 .....	5-3
5.1.1	Xt 関数によるロケールの設定 .....	5-3
5.1.2	Xt 関数におけるフォントセット・リソースの使用 .....	5-4
5.1.3	Xt ライブラリ関数におけるテキスト入力イベントのフィルタリング .....	5-4
5.1.4	Xt ライブラリ関数におけるロケールのコードセット・コンポーネントの取り込み .....	5-5
5.2	OSF/Motif および DECwindows Motif ツールキットの国際化機能の使用 .....	5-5
5.2.1	Motif アプリケーションにおける言語の設定 .....	5-5
5.2.2	コンパウンド・ストリングと , XmText および XmTextField ウィジェットの使用 .....	5-6
5.2.3	ウィジェット・クラスの国際化機能 .....	5-8
5.3	X ライブラリの国際化機能の使用 .....	5-8
5.3.1	ロケールの管理 .....	5-9
5.3.2	さまざまなロケールでのテキストの表示 .....	5-12
5.3.2.1	フォントセットの作成と操作 .....	5-13
5.3.2.2	フォントセットのメトリックスの取得 .....	5-15
5.3.2.3	フォントセットを使用したテキストの表示 .....	5-15
5.3.2.4	X 出力サーバを使用したテキスト処理 .....	5-18
5.3.2.5	エンコーディングが異なるフォントセット間の変換 ..	5-19
5.3.3	クライアント間通信の処理 .....	5-20
5.3.4	地域化されたリソース・データベースの処理 .....	5-22
5.3.5	X 入力サーバを使用したテキスト入力の処理 .....	5-23
5.3.5.1	入力サーバへの接続と , 接続の解除 .....	5-23
5.3.5.2	入力サーバ値の問い合わせ .....	5-25
5.3.5.3	入力サーバのコンテキストの作成と使用 .....	5-28

5.3.5.4	On-the-spot 入力スタイル用の前編集コールバックの作成 .....	5-31
5.3.5.5	入力サーバのためのイベント・フィルタリング .....	5-34
5.3.5.6	キーボードからのコンパウンド・ストリングの取得 ..	5-35
5.3.5.7	入力サーバが失敗したときの処理 .....	5-37
5.3.6	Xt および X ライブラリの国際化機能の使用法: 要約 .....	5-39

## 6 ロケールの作成

6.1	ロケールに対応した文字マップ・ソース・ファイルの作成 ...	6-2
6.2	ロケール定義ソース・ファイルの作成 .....	6-7
6.2.1	LC_CTYPE ロケール・カテゴリの定義 .....	6-9
6.2.2	LC_COLLATE ロケール・カテゴリの定義 .....	6-14
6.2.3	LC_MESSAGES ロケール・カテゴリの定義 .....	6-19
6.2.4	LC_MONETARY ロケール・カテゴリの定義 .....	6-21
6.2.5	LC_NUMERIC ロケール・カテゴリの定義 .....	6-24
6.2.6	LC_TIME ロケール・カテゴリの定義 .....	6-25
6.3	マルチバイト・コードとワイド文字コード間の変換を行うライブラリの構築 .....	6-29
6.3.1	必須メソッド .....	6-30
6.3.1.1	fgetws 関数用の __mbstopcs メソッドの作成 .....	6-30
6.3.1.2	getwc() 関数用の __mbtopc メソッドの作成 .....	6-33
6.3.1.3	fputws() 関数用の __pcstombs メソッドの作成 .....	6-38
6.3.1.4	__pctomb メソッドの作成 .....	6-40
6.3.1.5	mblen() 関数用のメソッドの作成 .....	6-40
6.3.1.6	mbstowcs() 関数用のメソッドの作成 .....	6-44
6.3.1.7	mbtowc() 関数用のメソッドの作成 .....	6-46
6.3.1.8	wcstombs() 関数用のメソッドの作成 .....	6-50
6.3.1.9	wctomb() 関数用のメソッドの作成 .....	6-53
6.3.1.10	wcswidth() 関数用のメソッドの作成 .....	6-56

6.3.1.11	wcwidth() 関数用のメソッドの作成 .....	6-59
6.3.2	オプションのメソッド .....	6-61
6.3.3	ロケールで使用するシェアード・ライブラリの作成 .....	6-62
6.3.4	ロケールに対応した methods ファイルの作成 .....	6-63
6.4	ロケールの作成とテスト .....	6-64
 <b>7 国際化アプリケーションのプログラミング上の注意</b>		
7.1	入力システムの選択 .....	7-2
7.2	ユーザ定義文字と語句入力の管理 .....	7-4
7.3	ロケールの指定によるソート順序の割り当て .....	7-6
7.4	英語以外の言語におけるリファレンス・ページの処理 .....	7-7
7.4.1	nroff コマンド .....	7-7
7.4.2	tbl コマンド .....	7-10
7.4.3	man コマンド .....	7-10
7.5	データ・ファイルのコードセットの変換 .....	7-11
7.6	中国語および韓国語の PostScript サポートでのフォント・レンダラの使用 .....	7-13
7.6.1	マルチバイト PostScript フォント用のフォント・レンダラの使用法 .....	7-14
7.6.1.1	2 バイト PostScript フォント用のフォント・レンダラのセットアップ .....	7-14
7.6.1.2	UDC フォント用のフォント・レンダラのセットアップ .....	7-15
7.6.1.3	TrueType フォント用のフォント・レンダラの使用 ...	7-16
 <b>A 国際化インタフェースの要約表</b>		
A.1	ロケール宣言 .....	A-1
A.2	文字分類 .....	A-1
A.3	大文字/小文字変換と汎用プロパティ変換 .....	A-4
A.4	文字照合 .....	A-5



A.5	言語と文化習慣によって異なるデータへのアクセス .....	A-5
A.6	日付/時刻の変換と書式付け .....	A-5
A.7	テキストの書き込みと読み取り .....	A-6
A.8	数値変換 .....	A-7
A.9	マルチバイト文字とワイド文字の変換 .....	A-7
A.10	入出力 .....	A-9
A.11	文字列処理 .....	A-10
A.12	コードセット変換 .....	A-12
 <b>B ユーザ定義文字データベースのセットアップと使用法</b>		
B.1	ユーザ定義文字の作成 .....	B-3
B.1.1	cedit のユーザ・インタフェース画面上での操作 .....	B-5
B.1.2	フォント・グリフの編集 .....	B-9
B.2	システム・ソフトウェアが使用する UDC サポート・ファイル の作成 .....	B-20
B.3	X11 や Motif アプリケーションで使用する UDC フォントの処 理 .....	B-23
B.3.1	fontconverter のコマンド・オプションの使用法 .....	B-24
B.3.2	出力ファイル・フォーマットの制御 .....	B-26
 <b>C プログラムでの DECterm ローカライゼーション機能の使用</b>		
C.1	DECterm ウィンドウにおける罫線の処理 .....	C-1
C.1.1	パターンに従った罫線の描画 .....	C-1
C.1.2	パターンによる罫線の削除 .....	C-4
C.1.3	領域内のすべての罫線の削除 .....	C-4
C.1.4	その他の DECterm エスケープ・シーケンス .....	C-5
C.1.5	DECterm が罫線をサポートしているかどうかの判定 .....	C-7
C.2	DECterm プログラミングにおける制約 .....	C-7
C.2.1	ダウンライン・ローダブル文字 .....	C-7
C.2.2	DRCS 文字 .....	C-7

## D サンプル・ロケールのソース・ファイル

D.1	文字マップ (charmap) のソース・ファイル .....	D-1
D.2	ロケール定義のソース・ファイル .....	D-7

## 用語集

## 索引

## 例

3-1	メッセージ・テキスト・ソース・ファイル .....	3-3
3-2	対話型操作によるメッセージ・カタログの生成 .....	3-22
5-1	X ウィンドウ・アプリケーションにおけるロケールの設定 ...	5-11
5-2	X ウィンドウ・アプリケーションにおけるフォントセットの作成と使用 .....	5-14
5-3	X ウィンドウ・アプリケーションにおけるテキストの表示 ...	5-16
5-4	X ウィンドウ・アプリケーションにおける他のクライアントとの通信 .....	5-21
5-5	X ウィンドウ・アプリケーションにおける入力サーバへの接続と接続の解除 .....	5-24
5-6	入力サーバがサポートするユーザ入力スタイルの取得 .....	5-26
5-7	X ウィンドウ・アプリケーションにおける入力コンテキストの作成と破棄 .....	5-28
5-8	X ウィンドウ・アプリケーションにおける前編集コールバックの使用 .....	5-31
5-9	X ウィンドウ・アプリケーションにおける入力サーバのためのイベント・フィルタリング .....	5-35
5-10	X ウィンドウ・アプリケーションにおけるキーボード入力の取得 .....	5-36
5-11	入力サーバが失敗したときの処理 .....	5-38
6-1	サンプル・ロケール用の charmap ファイル .....	6-2

6-2	マルチバイト・コードセット用の charmap ファイルの一部 ..	6-5
6-3	ロケール・ソース定義ファイルの構成 .....	6-7
6-4	LC_CTYPE カテゴリの定義 .....	6-9
6-5	LC_COLLATE カテゴリの定義 .....	6-14
6-6	LC_MESSAGES カテゴリの定義 .....	6-19
6-7	LC_MONETARY カテゴリの定義 .....	6-22
6-8	LC_NUMERIC カテゴリの定義 .....	6-25
6-9	LC_TIME カテゴリの定義 .....	6-26
6-10	ja_JP.sdeckanji ロケールのための __mbstopcs_sdeckanji メソッド .....	6-31
6-11	ja_JP.sdeckanji ロケールのための __mbtopc_sdeckanji メソッド .....	6-33
6-12	ja_JP.sdeckanji ロケールのための __pcstombs_sdeckanji メソッド .....	6-38
6-13	ja_JP.sdeckanji ロケールのための __pctomb_sdeckanji メソッド .....	6-40
6-14	ja_JP.sdeckanji ロケールのための __mblen_sdeckanji メソッド .....	6-41
6-15	ja_JP.sdeckanji ロケールのための __mbstowcs_sdeckanji メソッド .....	6-44
6-16	ja_JP.sdeckanji ロケールのための __mbtowc_sdeckanji メソッド .....	6-46
6-17	ja_JP.sdeckanji ロケールのための __wctombs_sdeckanji メソッド .....	6-51
6-18	ja_JP.sdeckanji ロケールのための __wctomb_sdeckanji メソッド .....	6-54
6-19	ja_JP.sdeckanji ロケールのための __wcswidth_sdeckanji メソッド .....	6-56
6-20	ja_JP.sdeckanji ロケールのための __wcwidth_sdeckanji メソッド .....	6-59

6-21	ja_JP.sdeckanji ロケールで使用されるメソッドのライブラリ の作成 .....	6-62
6-22	ja_JP.sdeckanji ロケールのための methods ファイル .....	6-63
6-23	fr_FR.ISO8859-1@example ロケールの作成 .....	6-64
6-24	LOCPATH 変数の設定とロケールのテスト .....	6-66
7-1	省略時の cp_dirs ファイル .....	7-4

## 図

3-1	メッセージ・カタログを使用できるようにするための、既存の プログラムの変換 .....	3-20
B-1	ユーザ定義文字をサポートするコンポーネント .....	B-3
B-2	cedit のユーザ・インタフェース画面 .....	B-5
B-3	credit フォント編集画面 .....	B-10
B-4	フォント・サイズを変更するフォント編集画面の解釈 .....	B-12
B-5	cedit の編集機能を呼び出すためのキーマップ .....	B-15
C-1	DECDRLBR シーケンスを用いた罫線の描画 .....	C-2
C-2	DECDRLBR パラメータのビット・パターン .....	C-3

## 表

3-1	メッセージ・テキスト・ソース・ファイルにおける特殊文字の 記述 .....	3-6
4-1	ワイド文字を追加し、カーソルを進める curses ルーチン .....	4-3
4-2	ワイド文字を挿入し、カーソルを進めない curses ルーチン ..	4-4
4-3	ワイド文字列を追加しカーソルを進めない curses ルーチン ..	4-5
4-4	ワイド文字列を追加しカーソルを進める curses ルーチン .....	4-6
4-5	ワイド文字列を挿入しカーソルを進めない curses ルーチン ..	4-8
4-6	ワイド文字列を削除する curses ルーチン .....	4-9
4-7	ウィンドウからワイド文字を読み取る curses ルーチン .....	4-9
4-8	ワイド文字列とその属性を読み取る curses ルーチン .....	4-10
4-9	属性なしでワイド文字列を読み取る curses ルーチン .....	4-12

4-10	ワイド文字列を端末から読み取るcursesルーチン .....	4-13
4-11	ワイド文字列をキーボードから読み取る curses ルーチン .....	4-14
4-12	ウィンドウ内の書式付きテキストを変換する curses ルーチン .....	4-15
4-13	ウィンドウに書式付きテキストを表示する curses ルーチン ..	4-16
5-1	X ライブラリのロケール宣言関数 .....	5-10
5-2	フォントセットの作成と操作を行う X ライブラリ関数 .....	5-13
5-3	テキストのサイズを調節する X ライブラリ関数 .....	5-15
5-4	テキストを表示するための X ライブラリ関数 .....	5-16
5-5	出力サーバと出力コンテキスト用の X ライブラリ関数 .....	5-18
5-6	クライアント間通信用の X ライブラリ関数 .....	5-20
5-7	地域化されたりソース・データベースを扱う X ライブラリ関 数 .....	5-22
5-8	入力コンテキスト (XIC) を管理する X ライブラリ関数 .....	5-31
7-1	英語でサポートされているコードセット変換 .....	7-13
7-2	UDC 文字用の XLFD 登録名 .....	7-16
B-1	UDC サポート・ファイルのオンデマンド・ローディングのた めの stty オプション .....	B-2
B-2	cedit コマンドのオプション .....	B-4
B-3	さまざまなフォント編集機能のためのキー .....	B-15
B-4	cedit モードを切り替えるためのキー .....	B-16
B-5	カーソル移動を制御するためのキー .....	B-16
B-6	ウィンドウ領域にカーソルを移動するためのキー .....	B-17
B-7	フォント・グリフを描画するためのキー .....	B-17
B-8	フォント・グリフを編集するためのキー .....	B-18
B-9	cgen コマンドのオプション .....	B-21
B-10	fontconverter コマンドのオプションと引数 .....	B-25
C-1	罫線に対する標準エスケープ・シーケンスの効果 .....	C-5



---

## まえがき

HP Tru64 UNIX では、国際化機能として、さまざまな国で利用できるプログラムを作成するためのツールとルーチン群を提供します。これらのツールやルーチンを使用することにより、次の特徴を持ったプログラムを作成することができます。

- どの国のユーザに対しても、そのユーザのために設計されたように見えるインタフェース
- 特定の言語や慣習に依存しないソース・コード

### 対象読者

本書は、多国語環境あるいは英語以外の言語環境で使用するプログラムの作成に経験のあるアプリケーション開発者を対象としています。また本書は、国際化プログラムで表示されるプログラム・メッセージを翻訳する方にも有用です。

### 新規機能と変更事項

本書は、Tru64 UNIX バージョン 5.1B 用に作成し直されました。Tru64 UNIX 『国際化ソフトウェア・プログラミング・ガイド』には、国際化アプリケーションのプログラミングに特有の情報が含まれるようになりました。Tru64 UNIX の国際化機能の使用についての情報は、姉妹編の『国際化機能ユーザズ・ガイド』に移されました。本書には、次の事項に関連した変更も盛り込まれています。

- 第2章がアップデートされ、Unicode ロケール、dense コード・ロケール、\*.UTF-8 ロケール、および ISO 8859-15 (Latin-9) ロケールや UTF-8 ロケールと追加のビットマップ・フォントを含むユーロ通貨記号の拡張サポートの情報が新しくなりました。
- 第3章に、翻訳可能なメッセージ・ファイルの書き方についての説明とガイドラインが追加されました。

## 本書の構成

本書は、以下のように構成されています。

第 1 章	各国のユーザのニーズに合ったプログラムを作成するための基本概念と手順について説明します。
第 2 章	国際化アプリケーションで、文字セット、文化的データ、および言語を扱うための手法について説明します。
第 3 章	メッセージを抽出して翻訳する方法と、メッセージ・カタログの作成およびアクセス方法について説明します。
第 4 章	ワイド文字データの書き込み、削除、および読み取りを行うための <code>curses</code> ライブラリ・ルーチンについて説明します。
第 5 章	国際化プログラムを作成する際に、GUI プログラミング・ライブラリ (X, OSF/Motif, および OSF/Motif に対する DECwindows 拡張機能) をどのように使用するかについて説明します。
第 6 章	ロケールのソース・ファイルと、ライブラリ・メソッドの作成方法、ロケールの構築およびテスト方法について説明します。
第 7 章	国際化アプリケーションの作成に関連する、プログラミング上のさまざまなトピックについて説明します。ここには、入力メソッド、ユーザ定義文字、ソート、リファレンス・ページの作成、データ・ファイルのコードセット変換、およびフォント・レンダラについての説明があります。
付録 A	ロケールの初期化、文字の分類、大文字と小文字の変換、文字の照合、日付と時刻の表記、テキスト文字列、数値の変換、マルチバイト文字、および文字列操作を実行する国際化関数とその要約を示します。
付録 B	中国語、日本語、韓国語のユーザ定義文字 (UDC) のサポートについて説明します ( <code>cedit</code> と UDC フォントの情報を含む)。
付録 C	DECterm のプログラミング機能と制限事項について説明します。
付録 D	第 6 章で説明しているサンプル・ロケールの全ソース・ファイルを示します。
用語集	本書で使用される用語と省略語を明確にします。



## 関連ドキュメントおよび標準

本書は、アプリケーション・プログラマから見た国際化に焦点を置いて説明します。姉妹編の『国際化機能ユーザズ・ガイド』は、国際化アプリケーションのユーザに焦点を置いています。このマニュアルは、オペレーティング・システムのドキュメンテーション・セットに含まれています。このマニュアルでは、さまざまな言語環境でアプリケーションを使用するための設定条件や、多国語作業環境でオペレーティング・システム・コマンドを使用する方法について説明しています。

Tru64 UNIX ドキュメント・セットに含まれる以下のマニュアルは、Tru64 UNIX システムで C コンパイラやその他のプログラム開発ツールを使用するための情報を提供します。国際化アプリケーションを開発するときは、これらのマニュアルを参照し、プログラミングに関する一般的な情報を入手してください。

- 『プログラミング・ガイド』
- 『プログラミング・サポートツール・ガイド』

Tru64 UNIX の『ドキュメント概要』には、このオペレーティング・システムに用意されているすべてのドキュメントについての情報があります。

Tru64 UNIX のドキュメントは次の URL でアクセスできます：

<http://tru64unix.compaq.co.jp/document/index.html>

次のマニュアルが、O'Reilly and Associates, Inc. から出版されています。

- 『*Programmer's Supplement for Release 6 of the X Window System*』

次の書籍が、共立出版より出版されています。

- 『国際化プログラミング - I18N ハンドブック』 (ISBN4-320-02904-6)

本書で説明するソフトウェア・コンポーネントは、以下の規格や規格案に準拠しています。本書では、これらの規格のいくつかを参照します。

- *ANS X3.159 Programming Language C*
- *ISO/IEC 646: 1983*  
情報処理 – ISO 7 ビット・コード化文字セットによる情報交換
- *ISO 6937: 1983*

情報処理 – コード化文字セットによるテキスト通信

- *ISO 8859-1: 1987*

情報処理 – ISO 8 ビット・シングルバイト・コード化グラフィック文字  
セット – Latin アルファベット No.1

- *ISO/IEC 9899: 1990*

情報科学技術 – プログラミング言語 – C

- *ISO/IEC 9945-1: 1990*

情報科学技術 – ポータブル・オペレーティング・システム・インタ  
フェース (POSIX) – Part 1: システム・アプリケーション・プログラミン  
グ・インタフェース (API) [C 言語]

- *ISO/IEC 9945-2: 1993*

情報科学技術 – ポータブル・オペレーティング・システム・インタ  
フェース (POSIX) – Part 2: シェルおよびユーティリティ

- *ISO/IEC 10646:2001*

情報科学技術 – UCS (Universal Multiple-Octet Coded Character Set)  
2001。この標準で定義される基本多言語プレーンは、Unicode 文字エン  
コーディングの本体と同じです。

- 情報交換用符号, *JIS X0201-1976*; 日本工業規格
- 情報交換用漢字符号, *JIS X0208-1990*; 日本工業規格
- 情報交換用漢字符号 – 補助漢字, *JIS X0212-1990*; 日本工業規格
- 中華人民共和国文字入力規格 (*GB18030-2000*); 中国国家規格, 北京, 2001。
- 中華人民共和国国家標準情報交換用漢字コード字符, 基本集 (*GB2312-80*); 中国国家文字セット規格, 北京, 1980
- 国家標準情報交換用文字集合, *CNS 11643*; 台湾, 1986, 1992
- 情報交換用韓国語文字集合規格, *KSC 5601*; 韓国, 1987
- タイ工業規格, *TIS 620-2533*; タイ語情報交換に使用されるグラフィック文字の主要セット規格
- The Open Group UNIX CAE 仕様。特に以下の各仕様。
  - *Commands and Utilities, XCU Issue 5*

- *Systems Interfaces and Headers, XSH Issue 5*
- *System Interface Definitions, Issue 5*
- *Networking Services, Issue 5*
- *X/Open Curses, XCURSES, Issue 4 Version 2*
- *The Unicode Standard, Version 3.0 and Version 3.1*
- *X11R6 Specification* (X の入力および出力メソッドを含む)

『*Programming for the World: A Guide to Internationalization*』(O'Donnell, Sandra Martin, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1994) では、世界中の文化および言語が持つ特性について紹介し、これらの特性をコンピュータ・システムで扱うために必要となる変更について説明しています。

『*Digital Technical Journal*』の Volume 5 Number 3 (1993 年夏発行) には、DEC 製品の国際化に関する記事が掲載されています。

## 表記法

本書では以下の表記法を使用しています。

%	パーセント記号は、C シェルのシステム・プロンプトを表します。
\$	ドル記号は、Bourne シェルと Korn シェルのシステム・プロンプトを表します。
#	番号記号は、スーパーユーザのプロンプトを表します。
% cat	対話型の例で使用されるボールド体は、ユーザ入力を示します。
<i>file</i>	イタリック体 (斜体) は、変数値、プレースホルダ、および関数の引数名を表します。
[   ]	構文中の大カッコはオプションの項目を表します。 大カッコ内の項目を区切る縦線は、リストから 1 つの項目を選択することを示します。

{|}

構文中の中カッコは必須項目を表します。中カッコ内の項目を区切る縦線は、リストから1つの項目を選択することを示します。

...

構文中の水平の反復記号は、先行する項目を1回以上繰り返して使用できることを示します。

:

垂直の反復記号は、例の一部が省略されていることを示します。

cat(1)

リファレンス・ページのクロス・リファレンスには、カッコで囲まれたセクション番号が含まれています。たとえば cat(1) は、cat コマンドの情報がリファレンス・ページのセクション1に記載されていることを示します。

Ctrl/x

この記号は、スラッシュの前のキーを押したまま、スラッシュの後のキーまたはマウス・ボタンを押すことを示します。このキーの組み合わせは、Ctrl/C のようにボックスで囲まれています。

Alt x

スペースで区切られた複数のキーまたはマウス・ボタンは、それぞれのキーを順に押すことを示します。例で使用されるときは、各キーはボックスで囲まれています。たとえば、AltQ。

## 国際化ソフトウェア開発の概要

プログラムが扱う言語や、文化的データ、文字のエンコード方式についてあらかじめ知らなくても、プログラムの開発が行える仕組みのことを国際化と呼びます。システムの観点からは、プログラムが実行時の環境に合わせ、さまざまな出力を生成できるようにするインタフェースを提供することを国際化といいます。国際化 (internationalization) を略して **I18N** と呼ぶことがあります。

本書では、国際化プログラムの開発に役立つ、Tru64 UNIX のインタフェースとユーティリティについて説明します。これらのインタフェースとユーティリティは、X/Open の UNIX 標準仕様に準拠しています。この標準仕様では、一部のインタフェースとユーティリティで、ベンダの実装に依存した動作が認められています。本書では、Tru64 UNIX に固有のソフトウェア特性については、その旨を明記します。

以下の節では、国際化プログラムの開発に使用されるオペレーティング・システム・インタフェースおよびユーティリティの概要について説明します。

- 言語。 1.1 節では、言語に依存する処理の実装について概要を説明します。 1.2 節では、ソフトウェア・アプリケーションの立場から言語を明確にします。
- 文化的データ。 1.3 節では、文化的データを明確にします。 1.1.1 項では、コンピュータ・システムでの文化的要件またはローカル要件の実装について説明します。
- 文字セット。 1.4 節では、国際化の立場から文字セットを明確にします。

### 1.1 言語宣言

言語宣言は、システム全体で、またはアプリケーションごと、あるいはユーザごとに、言語、文化的データ、およびコードセットの要件を設定するためのメカニズムです。言語宣言は、いくつかの予約されている環境変数にロケール名を設定することで行います。システム管理者は、シェル環境ごとにこれらの変数の省略値を設定できます。シェルに設定するロケールの省略値

については、Tru64 UNIX の『システム管理ガイド』を参照してください。  
ロケール変数は、プロセスごとに設定することもできます。

一般に、国際化プログラムは実行時にロケール変数を読み取り、ロケール変数の設定内容をプログラムの動作環境のロケール・カテゴリに取り込みます。ただし、プログラムは、そのつどこれらのカテゴリを内部的に設定できます。そのため、特定のロケールへのバインドは、プログラムのすべての部分に適用されるわけではありません。同一の実行サイクル内で、プログラムの部分ごとに、異なる地域化を適用することもできます。

### 1.1.1 地域化

コンピュータ・システムに、ある地域における要件を実装する仕組みのことを地域化といいます。これらの地域的要件のいくつかは、ロケールによって解決できます。各ロケールは、その地域の言語、文化的データ、およびコードセットの特定の組み合わせをサポートするデータの集合です。ロケールに含めることができる情報の種類と、ロケールを使用するインタフェースは、標準化されていますが、システム上のロケールのインストール位置とロケール名は、ベンダごとに異なります。

ロケールを提供するだけで地域化が実現できるわけではありません。たとえば、地域化を実装するには、翻訳したソフトウェア・メッセージを用意して、適切なフォントと単位系をサポートし、さらにそれらが表示デバイスと印刷デバイスで利用できるようにする仕組みが必要になります。また場合によっては、地域に固有な要件を満たすために、追加のソフトウェアを作成しなければなりません。

地域化 (localization) を略して **L10N** と呼ぶことがあります。

アプリケーション・プログラムでの地域化データおよびメッセージ・ファイルの作成と使用については、第 3 章を参照してください。地域化とグラフィカル・ユーザ・インタフェースについては、第 5 章を参照してください。

地域化を実現する際のプログラミングについては第 2 章を参照し、ロケール (ソフトウェアの地域化の主要なツール) については第 6 章を参照してください。

## 1.2 言語

国際化プログラムは、特定の言語で文字データ (テキスト) を扱うことはありません。

言語は、テキスト処理における文字の扱いや単語の順序に影響します。Tru64 UNIX には、国際化プログラムが、個々のユーザの使用する言語に応じてテキストを操作できるようにするインタフェースが用意されています。

言語の違いを吸収するためには、メッセージ・テキストとプログラム・コードを分離する必要があります。Tru64 UNIX には、メッセージ・テキストをコードから分離して、別の言語に翻訳し、実行時にプログラムからアクセスできるようにする機能が備わっています。第 3 章では、国際化インタフェース (WPI: **Worldwide Portability Interfaces**) を使用する国際化プログラムがどのようにメッセージを生成し、アクセスするかについて説明します。

X インタフェースと Motif インタフェースを使用する国際化プログラムは、次の方法でメッセージ・テキストをプログラム・コードから分離できます。

- UIL (ユーザ・インタフェース言語) ファイル内にメニュー項目、タイトル、テキスト・フィールド、およびメッセージを定義する。
- アプリケーションのリソース・ファイル内にタイトルとフォント・リストを指定する。
- ヘルプ・ウィジェットが使用するファイル内にヘルプ・メッセージを指定する。

X インタフェースと Motif インタフェースでメッセージ・テキストをプログラム・コードから分離する方法については、次のドキュメントを参照してください。

- 『*X Window System Toolkit*』
- 『*Common Desktop Environment: プログラマーズ・ガイド (国際化対応編)*』

### 1.2.1 文字分類

文字分類情報には、有効な各文字コードについての詳細な属性情報が含まれています。つまり、コードがアルファベット、大文字、小文字、句読点、制御文字、スペースなどの文字種を定義しているかどうかを示します。文

字分類関数と国際化正規表現は、どちらもこの情報を使用して文字クラスを判別します。

### 1.2.2 大文字と小文字の変換

大文字/小文字変換では、有効な各文字コードにおける大文字と小文字を識別するための情報が参照されます。大文字/小文字変換関数はこの情報を使用して、文字を大文字から小文字に、または小文字から大文字に変換します。すべての英字に大文字と小文字の区別があるわけではなく、また、まったく区別のない言語もあります。

### 1.2.3 メッセージ・カタログ

メッセージ・カタログは、特定の言語のためのプログラム・メッセージ、コマンド・プロンプト、およびプロンプトへの応答を含むファイルまたは記憶領域です。Motif アプリケーションでは、ロケールごとに異なるテキストやその他の値のために、メッセージ・カタログに加え、あるいはメッセージ・カタログの代わりに、リソース・ファイルと UIL ファイルを使用します。メッセージ・システムについては、第 3 章で説明します。

## 1.3 文化的データ

文化的データとは、地理上の領域 (本書では地域と呼びます) の慣習のことです。文化的データには、日付、時刻、通貨の書式などがあります。

国際化プログラムは、これらの文化的データがどのように設定されているかをあらかじめ想定することは不可能なため、実行時にシステム機能を使用して書式を判定します。この機能は、言語情報データベース (**langinfo** データベース) から提供されます。プログラムは、文化的データ項目に必要な書式情報をこのデータベースから取り出します。

### 1.3.1 言語情報

ロケールごとに異なる文化的データの書式と設定を記述する地域化データのことを、言語情報といいます。langinfo データベースに格納される情報には、日付、時刻、通貨、および数値のための適切な書式と文字が含まれます。



## 1.4 文字セット

文字セットとは、自然言語やコンピュータ言語の単語と他の基本単位を構成する、アルファベットやその他の文字の集合です。コード化文字セット(コードセット)とは、文字セットを定義し、その文字セット中の文字とそのビット表現との1対1の対応を明確に定義する規則の集合です。

さまざまなコードセットで記録されているテキストを扱うプログラムでは、文字コードのサイズやビット割り当てについて前提を設けることはできません。特にそのようなプログラムでは、文字が格納されている領域の一部を他の用途に利用することはできません。

### 1.4.1 照合順序

照合順序(文字の順序付け)は、ハードウェアでは暗黙的に決まっていることがあります。ソフトウェアでは特定の地域で使用される言語に合わせて定義できます。多くの言語には、複雑なソート規則があります。照合規則の例を次に示します。

- 1つのアルファベットが1つの文字で表現されるとはかぎらない。  
たとえば伝統的なスペイン語では、`ch`という文字の組み合わせが、文字`c`と`d`の間にソートされます。
- 1つの文字が、複数文字の組み合わせと等価なことがある。  
たとえば、`ß`という文字は、標準ドイツ語とスイス・ドイツ語では`ss`と等価であり、オーストリア・ドイツ語では`sz`と等価です。
- アクセント文字が、アクセントの付かない文字の後に来るとはかぎらない。  
多くの言語では、単語に含まれる文字がアクセント文字以外すべて同じ場合にのみ、アクセント文字を含む単語が後に置かれます。また、特定のアクセント文字を独自の文字と見なし、アクセントの付かない文字とは別の文字の後にソートする言語もあります。
- 同じ言語でも、文字が複数の方法でソートされることがある。  
アジア系言語の表意文字には、読みに基づくソート順序と、2種類の視覚的な構成要素(意味を表す要素である部首と、画数)に基づくソート順序があります。

各ロケールには、対応するコードセットで定義されている文字の相対順序に関する照合順序情報が含まれており、この情報は文字列比較関数に渡されます。また国際化対応の正規表現も、この照合順序を使用して、文字範囲、照合シンボル、および等価クラスを実装します。

## 1.4.2 文字と文字列

文字は、1つのグラフィック・シンボルまたは制御コードを表す、1つ以上のバイトの並びです。文字という用語を、Cプログラミング言語のデータ型である `char` と混同しないようにしてください。`char` は、基本実行文字セットの任意のメンバを格納するのに十分な大きさで、通常は8ビット値にマップされるオブジェクトを表します。Cの `char` データ型とは異なり、文字は、1バイトまたは複数バイトの値で表現されます。マルチバイト文字という表現は、文字という用語と同義です。つまり両者とも、1バイト値を含む、任意の長さの文字値を意味します。

文字の列または文字列は、ヌル・バイトで終わる、連続したバイト(ヌル・バイトも含む)の並びです。Cプログラミング言語では、文字列は `char` 型の配列です。ヌル・バイトは、すべてのビットにゼロ(0)が設定されている値です。

ワイド文字は、拡張実行文字セットの任意のメンバを格納するのに十分な大きさの整数型です。プログラムの観点からは、ワイド文字は、ヘッダ・ファイル `/usr/include/stddef.h` (X/Open XSH Specification に準拠) および `/usr/include/stdlib.h` (ANSI C 標準に準拠) で定義されている `wchar_t` 型のオブジェクトです。これらのヘッダ・ファイルの位置は、標準化組織が決定しますが、定義そのものは実装によって異なります。たとえば、1バイト・コードセットだけをサポートする実装(Tru64 UNIX には当てはまりません)では、`wchar_t` を1バイト値として定義することもあります。

ワイド文字列は、ヌル・ワイド文字で終わる、連続したワイド文字(ヌル・ワイド文字も含む)の並びです。ワイド文字列は、`wchar_t` 型の配列です。ヌル・ワイド文字は、すべてのビットにゼロ(0)が設定されている `wchar_t` 値です。

空の文字列は、1番目の要素がヌル・バイトの文字列です。同様に、空のワイド文字列は、1番目の要素がヌル・ワイド文字のワイド文字列です。

### 1.4.3 ポータブル文字セット

ポータブル文字セット (PCS) は、すべてのロケールにおいて、コンパイル時 (ソース) と実行時 (実行可能) 環境の両方でサポートされます。PCS には次の文字が含まれます。

- 英語のアルファベットの 26 個の大文字

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- 英語のアルファベットの 26 個の小文字

a b c d e f g h i j k l m n o p q r s t u v w x y z

- 10 個の 10 進数

0 1 2 3 4 5 6 7 8 9

- 次の 32 個のグラフィック文字

! " # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~

- スペース文字と、水平タブ、垂直タブ、および書式送りを表す制御文字
- 上記の文字に加え、実行時環境の PCS には、アラート、後退、復帰、および改行を表す制御文字が含まれる。

X/Open が規定するポータブル文字セットは、*ISO/IEC 9899: 1990* で定義されている基本ソース文字セットおよび基本実行文字セットに類似しています。ただし、X/Open バージョンには、ドル記号 (\$)、アットマーク (@)、およびグレイブ・アクセント ( ` ) も含まれます。

一部のロケール (たとえば、**ISO 646** の修正バージョン) では、上記の文字のいくつかは、他の文字で代用されていることがあります。そのような場合、代用文字は、ポータブル文字セット中で置き換えられた文字と構文上同じ意味を持ちます。文字の代用の例としては、省略時の番号記号 (#) に対するイギリスのポンド記号 (£) があります。

すべてのコードセットに対応するポータブルな文字セットが定義された背景には、エンコード方式でサポートできる言語の数が限られていることが関係しています。一般に、このことは、UNIX システム用に開発された文字エンコード方式にも当てはまります。言い換えれば、中国語ロケールで使用されるコードセットは、中国語の文字に加え、すべての PCS 文字を含んでいなければなりませんが、ロシア語やアイスランド語をサポートするのに必要な文字を含むことはありません。同様に、ロシア語に使用されるコードセットは、中国語の文字を含むことはありませんが、すべての PCS 文字を含ん

でなければなりません。つまり、ロケール設定が何であっても、プログラムは PCS 文字を使用できることになります。

#### 1.4.4 ユニバーサル文字セット

Unicode および ISO/IEC 10646 標準で規定されているユニバーサル文字セット (UCS) は、本オペレーティング・システムでサポートされています。UCS は、すべての主な言語で利用できる文字の種類を規定し、文字を処理するために各言語で使用するルールを標準化します。この文字セットは、アプリケーションが同じエンコード方式と規則を使用することにより、すべての言語の文字を操作できるようにするという考えに基づいています。このため、オペレーティング・システムで UCS をサポートするときには、複数のアルゴリズムを持つことなく、ファイル・コードと国際化プロセス・コードの変換をサポートすることができます。

ISO/IEC 10646 標準は、データの入出力に異なる解析方式を必要とする、2 種類の文字サイズ (16 ビットと 32 ビット) をサポートしています。16 ビット単位 (2 オクテット) で解析する UCS エンコーディングを、UCS-2 といいます。32 ビット単位 (4 オクテット) で解析する UCS エンコーディングを、UCS-4 といいます。UCS-4 は、より多くの文字をサポートでき、大規模なコンピュータ・システムの内部処理コードとして使用したときに、効率的に操作することができます。

ファイル・データを取り扱うバイト指向のプロトコルで使用する UTF (Universal Transformation Format) を、標準では何種類も規定しています。本オペレーティング・システムでは、UTF-8 と UTF-32 の使用をお勧めします。本オペレーティング・システムは、次の UTF をサポートしています。

- UTF-8

UTF-8 は、UCS-4 の処理コードを 8 ビットのバイト・シーケンスに変換し、C0 コード・ポジション (0 ~ 31)、SPACE 文字 (32)、および DEL 文字 (127) の変換の透過性を保証する、標準的な方式です。本オペレーティング・システムには、UTF-8 用のコードセット・コンパータおよびロケールが用意されています。

- UTF-16

UTF-16 は、Unicode 標準で規定されている代替文字拡張手法を使用し、16 ビット単位で文字を表現します。UTF-16 は UCS-2 のスーパーセットですが、UCS-4 のコード空間のすべての表現はサポートしていま

せん。UTF-16 は、両方の標準でカバーしている言語で現在定義されている文字をすべてサポートしているわけではありません。

ファイル・コードのバイト・オーダーは、ファイルが生成されたプラットフォームにより異なり、リトル・エンディアン (LE) またはビッグ・エンディアン (BE) になります。UTF-16 では、バイト・オーダー・マーク (BOM) を使用して、バイト・オーダーを示します。BOM は、ファイル・テキスト・データの一部ではありません。Unicode 標準には、BOM を含まない UTF-16LE と UTF-16BE (それぞれリトルエンディアンとビッグエンディアンのバイト・オーダー) も規定されています。本オペレーティング・システムは、コードセット・コンバータを通して、UTF-16、UTF-16LE、および UTF-16BE をサポートします。UCS-2 は UTF-16 のサブセットであるため、本オペレーティング・システムは、UCS-2 を UTF-16 コードセット・コンバータでサポートします。UCS-2 コードセット・コンバータ名は、UTF-16\* という別名で認識されます。ただし、文字の種類が制限されています。

オペレーティング・システムは通常、UTF-16LE や UTF-16BE ではなく、UTF-16 を想定します。入力時、システムは BOM を探します。BOM が見つからない場合、コンバータは UTF-16BE と見なします。出力時、システムは自動的に BOM を挿入します。アプリケーションが入力や出力時にリトル・エンディアンやビッグ・エンディアンのバイト・オーダーを前提とする場合は、バイト・オーダーを明示的に指示しなければなりません。バイト・オーダーの設定についての詳細は、`iconv_intro(5)` を参照してください。

- UTF-32

UTF-32 では、4 バイトのエンコーディング単位で文字が表現されます。UTF-32 は、文字の値の範囲が UTF-16 と同じ U+0000 ~ U+10FFFF に限定されている、UCS-4 の限定サブセットです。U+10FFFF を超えるプライベート用範囲は、ISO/IEC 10646 と Unicode 標準のエンコーディング・フォーマットの相互動作性を向上するために、ISO/IEC 10646 の将来のバージョンでは削除されます。

UTF-32 では BOM を使用して、バイト・オーダーがリトル・エンディアンとビッグ・エンディアンのどちらであることを示します。UTF-16 と同様に、Unicode 標準では BOM を含まない UTF-32LE と UTF-32BE を規定しています。オペレーティング・システムでの入力および出力の省略時の設定も、UTF-16 と同じです。

UCS-4 コードセット・コンバータを使用して、UTF-32 を処理します。

本オペレーティング・システムは、コードセット・コンバータとロケールによって、UCS-4 をサポートします。ロケールと一部のライブラリ関数では、アプリケーションが UCS-4 を内部処理コードとして使用することができます。コードセット・コンバータを使用すると、ファイル・データを、システムに常駐しているフォントと他のソフトウェアがサポートするエンコーディング・フォーマットに変換することができます。

Unicode、ロケール、および関連するエンコーディング・フォーマットについての詳細は、2.2 節を参照してください。curses ライブラリと、ワイド文字フォーマットおよびマルチバイト文字のサポートについての情報は、第 4 章を参照してください。

## 国際化ソフトウェアの開発

この章では、地域化の要件 (言語、コードセット、および文化習慣の違い) によって、基本的なコーディング作業がどのように変わるかを説明します。この章で説明するプログラム開発技術を適用するサンプル・アプリケーションが、ディレクトリ `/usr/examples/i18n/xpg4demo` にあります。このディレクトリにある `README` ファイルには、アプリケーションの紹介と、さまざまなロケールでこのアプリケーションをコンパイルして実行する方法についての説明が記載されています。本書では、アプリケーション・プログラム `xpg4demo` の一部を例として使用します。

多くのコンピュータ・プログラムの主要な機能の 1 つとして、データの操作が挙げられます。データ操作には、プログラムとコンピュータ・ユーザ間のやり取りが伴うことがあります。商用アプリケーションでは、このようなやり取りを各ユーザの母国語で行うことが重要です。文化的データも、適切な慣習に従っていなければなりません。

各国語でのデータ操作をサポートするプログラムを作成する場合、コンピュータ・システム内では、言語は 1 つまたは複数のコードセットで表現されることがある点を考慮しておかなければなりません。言語ごとにその要件が異なるため、コードセットの文字は、サイズ (8 ビット、16 ビットなど) とバイナリ表現の両方で異なることがあります。

言語や、文化的データ、文字のエンコーディングにまったく依存しないプログラムを作成することにより、上記のコードセットとデータの要件を満たすことができます。そのようなプログラムは、国際化されたプログラムと呼ばれます。サポートされている言語、地域、およびコードセットの組み合わせに固有のデータは、プログラム・コードとは別に保持され、言語初期化関数を使用して実行時環境にバインドできます。

Tru64 UNIX は、国際化ソフトウェアの開発、地域化データの定義、および特定の言語要件の宣言に使用する、次の機能を備えています。

- 各言語の言語、コードセット、および文化的データの定義を含むロケール (2.1 節)

- 拡張文字コードを処理し，言語とコードセットに依存しない文字分類，大文字/小文字変換，数値書式の変換，および文字列照合を行うライブラリ関数 (2.2 節)
- プログラムが文化的データや言語固有のデータを動的に判別できるようにするライブラリ関数 (2.3 節)
- プログラム・メッセージをプログラム・コードとは別に保持し，異なる言語に翻訳して，実行時にプログラムから取り出せるようにするメッセージ・システム (2.4 節)
- 各ユーザの言語および文化的要件を満たすように，プログラムを実行時にバインドする初期化関数 (2.5 節)

この章の説明と例では，標準 C ライブラリに含まれる関数に焦点を当てます。curses ライブラリの関数の使用方法については，第 4 章を参照してください。X および Motif ライブラリの関数の使用方法については，第 5 章を参照してください。

## 2.1 ロケールの使用

本オペレーティング・システムでは，**Unicode** ロケールと **dense** コード・ロケールがサポートされています。Unicode ロケールは，`/usr/i18n/lib/nls/ucslloc/` にインストールされます。dense コード・ロケールは，`/usr/i18n/lib/nls/loc` にインストールされます。省略時のロケールは，シンボリック・リンク `/usr/i18n/lib/nls/dloc` によって決まります。たとえば，日本語のロケール・ファイル名 `/usr/lib/nls/loc/ja_JP.eucJP` は，`/usr/i18n/lib/nls/dloc/ja_JP.eucJP` へのシンボリック・リンクです。ここで，`/dloc` は，日本語ロケールの Unicode バージョン用の `/ucslloc` と，dense コード・バージョン用の `/loc` のいずれかです。

スーパーユーザの場合，シンボリック・リンクの設定を変えることで，Unicode ロケールと dense コード・ロケールの間で切り替えることができます。`l10n_intro(5)` を参照するか，SysMan Menu から「国際化ソフトウェアの構成」ユーティリティを使用してください。このユーティリティを使用して，省略時のシステム・ロケールを変更し，複数の入力システムをサポートするアジア系言語のロケールでの入力システムを指定することもできます。詳細については，「国際化ソフトウェアの構成」ユーティリティのオンライン・ヘルプを参照してください。



Unicode ロケールは、Unicode と ISO/IEC 10646 標準に適合しており、UTF-32 をワイド文字エンコーディングとして使用します。UTF-32 ワイド文字エンコーディングでは、`wchar_t` の値はロケールに関係なく同じ文字を表します。これは、Unicode 標準では、プラットフォームが異なっても実装の一貫性が保たれるためです。

名前が `.UTF-8` で終わるロケールでは、ファイル・コードと、**ISO 10646** および Unicode 標準で規定された UTF-32 内部処理コード (`wchar_t` エンコーディング) を使用します。

その他の非 UTF-8 Unicode ロケールでは、内部処理コードとして UTF-32 を使用する一方、従来の UNIX コードセットや独自のコードセットをファイル・コード用に使用します。これらの Unicode ロケールのサブセットには、`@ucs4` という修飾子が付きます。ただし、これらのサブセットは、`@ucs4` 修飾子が付かないロケールと同じです。`@ucs4` サブセットは、旧製品との互換性を保つために用意されており、将来は削除される可能性があります。`@ucs4` ロケールは、CDE のログイン・メニューから選択することはできません。ロケール名を `LANG` 環境変数で指定しなければなりません。

`universal.UTF-8` ロケールを利用することもできます (エンド・ユーザーではなく、アプリケーション用)。このロケールは、UCS (Universal Character Set) の文字を完全にサポートしています。

エンコーディング・フォーマットについての詳細は、Unicode(5) のリファレンス・ページを参照してください。

`.UTF-8` ロケールでは、1 バイトを超えるコードの文字がファイル・コードに含まれることがあります。このため、これらのロケールは、マルチバイト・データを処理できるアプリケーションで使用してください。新しいアプリケーションは、マルチバイトの `.UTF-8` ロケールをベースにして設計してください。このロケールには多くの種類の文字が盛り込まれているため、将来、アプリケーションは文字セットを変更することなく文字サポートを拡張することができます。

`dense` コード・ロケールでは、テーブル・サイズを最小にするために、ワイド文字のエンコーディングに `dense` コードを使用します (つまり、コードポイントが連続して割り当てられ、空になる位置はありません)。`dense` コード・ロケールでは、1 つのロケールの `wchar_t` 値は、他のロケールで同じ文字を表しているとは限らず、ロケールに固有です。`dense` コード・ロケールは、内部処理コードに依存しないアプリケーションや、性能を上げることに

重点を置いているアプリケーション (dense コード・ロケールは Unicode ロケールよりも多少効率的であるため) に適しています。

マルチバイト文字セットの有効なすべてのコードポイントは、Unicode のプライベート用領域のコードポイントにマッピングされる対応先なしのコードポイントも含め、Unicode 内の有効なコードポイントにマッピングされます。このため、dense コード・ロケールは、Unicode ロケールと等価です。一般的に、Unicode ロケールと dense コード・ロケールで、同じ文字マップとロケール・ソースを使用することができます。ただし、LC\_COLLATE セクションで定義されていない Unicode 文字と dense コード文字は、異なる方法でソートされることがあります。

各 dense コード・ロケールに対して、Unicode ロケールが 1 つ存在します (ただし、すべての Unicode ロケールに dense コード・バージョンが存在するわけではありません)。Latin-1 ロケール (ISO8859-1) では、Latin-1 の文字が Unicode の最初の 256 文字と同じであるため、dense コード・ロケールと Unicode ロケールは全く同じです。シンボリック・リンクの設定に従って、同じロケール名で Unicode ロケールを参照する場合と、dense コード・ロケールを参照する場合があることに注意してください。このため、あるロケールでのアプリケーションの実行に問題がある場合、シンボリック・リンクをチェックしてください。

Unicode ロケールでは `wchar_t` フォームの文字に一貫した値を使用するため、Unicode ロケールにリンクすると、ロケールやプラットフォームの間で一貫性が増します。ただし、ユーザによっては文字を `wchar_t` フォームに変換する独自のアルゴリズムを使用する古い dense コード・ロケールを好む場合や、アプリケーションが dense コードの `wchar_t` エンコーディングに依存している場合があります。

## 2.2 コードセットの使用

以前は、大半の UNIX システムが 7 ビットの **ASCII** コードセットに基づいていました。しかし、英語以外のほとんどの言語は、ASCII コードセット以外の文字も含んでいます。

X/Open の UNIX 標準では、オペレーティング・システムが ASCII コードセットに加え、特定のコードセットもサポートすることは規定していません。ただし、この仕様は、システム上で利用可能なコードセットが何であ

れ、プログラムがそのコードセットの文字を処理できるようにするための、文字操作のインタフェースを必須要件として規定しています。

ISO (国際標準化機構) コードセットの 1 番目のグループは、主要なヨーロッパ言語だけをカバーしています。このグループ内のいくつかのコードセットでは、主要な言語を単一のコードセットに統合することができます。これらのコードセットはすべて、ASCII コードセットのスーパーセットであるため、既存の国際化されていないソフトウェアで問題を生じることはなく、英語以外の言語もサポートできます。Tru64 UNIX オペレーティング・システムは、**ISO 8859-1** (ISO Latin-1) を使用する、米国英語ロケールを常にサポートします。

WLS (ワールドワイド言語サポート) の一部としてインストールされているサブセットは、地域化に対応しているオペレーティング・システムをサポートし、上記以外の ISO コードセットに基づくロケールを含むことがあります。たとえば、チェコ語、ハンガリー語、ポーランド語、ロシア語、スロバキア語、およびスロベニア語をサポートする、Tru64 UNIX のオプションの言語別サブセットには、ISO 8859-2 (Latin-2) に基づくロケールが含まれています。

以下に、WLS が提供する ISO コードセットの全リスト、それらがサポートする言語、詳しい説明の記載されているリファレンス・ページを示します。

- ISO 8859-1, Latin-1

カタロニア語、デンマーク語、オランダ語、英語/イギリス、英語/アメリカ合衆国、フィンランド語、フラマン語/ベルギー、フランス語/ベルギー、フランス語/カナダ、フランス語/スイス、フランス語、ドイツ語/スイス、ドイツ語/ドイツ、アイスランド語、イタリア語、ノルウェー語、ポルトガル語、スペイン語、およびスエーデン語を含む、西ヨーロッパおよび北アメリカの言語

iso8859-1(5) のリファレンス・ページを参照してください。

- ISO 8859-2, Latin-2

チェコ語、ハンガリー語、ポーランド語、スロバキア語、およびスロベニア語を含む、東ヨーロッパの言語

iso8859-2(5) のリファレンス・ページを参照してください。

- ISO 8859-4, Latin-4

リトアニア語

iso8859-4(5) のリファレンス・ページを参照してください。

- ISO 8859-5, Latin/Cyrillic

ロシア語

iso8859-5(5) のリファレンス・ページを参照してください。

- ISO 8859-7, Latin/Greek

ギリシャ語

iso8859-7(5) のリファレンス・ページを参照してください。

- ISO 8859-8, Latin/Hebrew

ヘブライ語/イスラエル (ISO Hebrew コードセットを使用)

iso8859-8(5) のリファレンス・ページを参照してください。

- ISO 8859-9, Latin-5

トルコ語

iso8859-9(5) のリファレンス・ページを参照してください。

- ISO 8859-15, Latin-9

カタロニア語/スペイン, デンマーク語, オランダ語, 英語/イギリス, 英語/アメリカ合衆国, フィンランド語, フラマン語/ベルギー, フランス語/ベルギー, フランス語/カナダ, フランス語/スイス, フランス語, ドイツ語/スイス, ドイツ語/ドイツ, アイスランド語, イタリア語, ノルウェー語, ポルトガル語, スペイン語/スペイン, およびスウェーデン語。ISO 8859-15 (および UTF-8) は, ユーロ通貨文字をサポートしています。

iso8859-15(5) のリファレンス・ページを参照してください。

本オペレーティング・システムは, ISO 8859-3 (Latin-3) コードセットと ISO 8859-6 (Latin-6) コードセットはサポートしていません。

標準のオペレーティング・システム上のユーティリティがサポートする ISO コードセットには, 他に **ISO 6937: 1983** があります。このコードセットは, 7 ビットと 8 ビットの両方の文字に対応しており, 通信ネットワークや交換メディア (磁気テープやディスクなど) 上でのテキスト通信に使用されます。

ここまでで説明したコードセットには, 文字が 1 バイトに格納されるという要件があります。このようなコードセットは, 文字が複数バイトに格納されるアジア系言語の要件は満たしません。本オペレーティング・システムで

は、アジア系の言語および国をサポートするサブセットをインストールすることによって、次のコードセットが使用できます。

- 日本語
  - 日本語拡張 UNIX コード (省略時のコード)  
eucJP(5) を参照してください。
  - シフトJIS  
shiftjis(5) を参照してください。
  - DEC Kanji  
deckanji(5) を参照してください。
  - Super DEC Kanji  
sdeckanji(5) を参照してください。
- 韓国語
  - DEC Korean  
deckorean(5) を参照してください。
  - 韓国語拡張 UNIX コード  
eucKR(5) を参照してください。
- タイ
  - タイ API コンソーシアム/タイ工業規格  
TACTIS(5) を参照してください。
- 簡体字中国語
  - DEC Hanzi  
dechanzi(5) を参照してください。
  - GBK および GB18030  
GBK(5) と GB18030(5) を参照してください。
- 繁体字中国語
  - DEC Hanyu  
dechanyu(5) を参照してください。
  - 台湾拡張 UNIX コード

eucTW(5) を参照してください。

- BIG-5 (および改良版の Shift BIG-5)

    big5(5) および sbig5(5) を参照してください。

- Telecode

    telecode(5) を参照してください。

これらのコードセットは、オペレーティング・システム・ソフトウェアのアジア言語サブセットをインストールすると、使用できるようになります。また、アジア言語文字の入力や表示に必要な専用の端末ドライバと関連ユーティリティも使用できるようになります。

PC システム用に開発されたコードセットは、一般にコード・ページと呼ばれます。UNIX システム用に開発された大半の言語固有のコードセットには、対応する PC コード・ページがあります。ほとんどの場合、Tru64 UNIX はファイル・データのあるエンコード方式から別のエンコード方式に変換するコンバータを介して、PC コードセットをサポートします。CP850 コードセットは、英語/アメリカ合衆国をサポートします。そして、文字のエンコーディングに CP850 コード・ページを使用して PC 上で生成されるアクセント付き文字を含むデータで使用されます。ヨーロッパの MS-DOS および Windows オペレーティング・システムでは、特に指定しなければ、通常この文字エンコーディングが使用されます。code\_page(5) のリファレンス・ページを参照してください。

Unicode と ISO/IEC 10646 標準は、ユニバーサル文字セット (UCS) を規定します。UCS はアジア系言語も含め、すべての言語に同じ規則を適用することにより、文字ユニットを扱えるようにする文字セットです。オペレーティング・システムは、この文字の UCS-4 (32 ビット) エンコーディングを処理コードでサポートします。

Unicode あるいは ISO/IEC 10646 標準 (または両方) で定義されている他のエンコーディング・フォーマットには、次のものがあります。

- UCS-4 の 16 ビット版である UCS-2
- UCS エンコーディングを、バイト指向プロトコルで取り扱うバイトのシーケンスに変換する各種の Universal Transformation Format (UTF-8, UTF-16, および UTF-32)

オペレーティング・システムでは、ロケールやコードセット・コンバータ (またはその両方) によって、これらの異なるフォーマットをサポートしてい

ます。UCS-2 は UTF-16 のサブセットであるため、本オペレーティング・システムは、UCS-2 を UTF-16 コードセット・コンバータでサポートします。本オペレーティング・システムは、コードセット変換とロケールの両方で UCS-4 をサポートします。

以下のロケールでは、UTF-32 を内部処理コードとして使用します。

- `universal.UTF-8`

このロケールはアプリケーションで使用し、UTF-8 ファイル・フォーマットのデータを UCS-4 処理コードに変換したり、UCS-4 文字をテストして LC-CTYPE クラス (`alnum`, `alpha`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, または `xdigit`) に含まれているか調べます。このロケールでは、`LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, および `LC_TIME` の定義が、POSIX (C) ロケールでの定義と一致します。アプリケーションでは、このロケールを `fold_string_w()` 関数とともに使用して、Unicode および ISO/IEC 10646 標準で定義されている文字をすべて処理することができます。

このロケールは、ローカルな文化的慣習にアクセスできないため、他の大半のロケールとは異なります。

- `language_territory.UTF-8`

これらのロケールは、分類情報を特定の言語の文字に制限し、アプリケーションで国ごとに異なるデータを利用できるようにします。また、ファイル・データが UTF-8 エンコーディング規則に従っていることを前提とします。ユーロ通貨記号をサポートする、本オペレーティング・システムのロケールは、UTF-8 コードセットまたは ISO8859-15 コードセットを使用します。

Unicode UTF-8 コードセットは、カタロニア語/スペイン、チェコ語、デンマーク語、オランダ語、英語/イギリス、英語/アメリカ合衆国、フィンランド語、フラマン語、フランス語/ベルギー、フランス語/カナダ、フランス語/スイス、ドイツ語/スイス、ドイツ語、ギリシャ語、ハンガリー語、アイスランド語、イタリア語、日本語、韓国語、リトアニア語、ノルウェー語、ポーランド語、ポルトガル語、ロシア語、スロバキア語、スロベニア語、スペイン語、スウェーデン語、トルコ語、簡体字中国語 (Hanzi)、繁体字中国語 (Hanyu) をサポートしています。Unicode(5) のリファレンス・ページを参照してください。

- `native_locale_name`

これらのロケールは、UTF-32 を内部処理コードとして使用します。  
`native_locale_name` のコードセット部分 (たとえば、ISO8859-1) は、ファイル・コードを指定します。また、ロケールには各国の言語の文字の分類情報が用意されていますが、UTF-32 のすべての文字の分類情報が用意されているわけではありません。アプリケーションでは、国固有の情報を利用できます。`LC_COLLATE`、`LC_MESSAGES`、`LC_MONETARY`、`LC_NUMERIC`、および `LC_TIME` のカテゴリ定義は、`native_language_name` の定義と一致します。

- `native_language_name@ucs4`

これらのロケールは、`@ucs4` ロケールを使用する既存のアプリケーションとの互換性を保つために用意されています。これらのロケールは、`native_locale_name` ロケールと同様に機能しますが、用意されているロケールのリストは、`native_language_name` ロケールほど完全ではありません。

`LC_TIME` などのロケール・カテゴリについては、2.5 節を参照してください。データ処理でのロケールと比較については、Unicode(5) のリファレンス・ページと 2.1 節を参照してください。ユーロ通貨記号についての詳細は、euro(5) のリファレンス・ページを参照してください。

UCS-2、UCS-4、UTF-8、UTF-16、および UTF-32 のサポートについての詳細は、Unicode(5) のリファレンス・ページを参照してください。特定地域の言語でコードセットがどのようにサポートされるかについては、その言語のリファレンス・ページを参照してください。言語のリファレンス・ページ、特にアジア系言語のリファレンス・ページには、ロケールではサポートされていないが、コードセット・コンバータが利用できる追加のコードセットについての説明も含まれています。

これ以降の項では、複数のコードセットの文字を処理するプログラムを作成する際に、ソース・コードの記述方法に影響する事柄について説明します。

- データの透過性の確保 (2.2.1 項)
- コード内リテラルの使用 (2.2.2 項)
- マルチバイト文字の操作 (2.2.3 項)
- マルチバイト文字データとワイド文字データ間の変換 (2.2.4 項)
- ソースおよび実行コードセットにおけるマルチバイト文字の規則 (2.2.5 項)



- 文字の分類 (2.2.6 項)
- 文字の変換 (2.2.7 項)
- 文字列の比較 (2.2.8 項)

### 2.2.1 データの透過性の確保

2.2 節で説明したように、国際化ソフトウェアは、さまざまな文字エンコード方式に対応しなければなりません。プログラムは、X/Open UNIX CAE Specifications の必須要件に準拠するすべてのシステム上で特定のコードセットがサポートされていることや、個々の文字のビット数が固定されていることを前提にはできません。

UNIX システムでは 7 ビットの ASCII 文字を長年にわたって使用してきたため、バイトの最上位ビット (MSB) を独自の用途に使用するプログラムが存在します。これは、基本コードセットの文字が常にバイトの下位 7 ビットにマップされている場合には安全な手法ですが、プログラミング手法としては好ましくありません。国際化コードセットの世界では、バイトの最上位ビットをプログラム独自の用途に使用することは避けなければなりません。

### 2.2.2 コード内リテラルの使用

国際化ソフトウェアを作成するときは、コード内リテラルを使用しないでください。例として、次の条件文について考えてみます。

```
if ((c = getchar()) == \141)
```

この条件文では、小文字 `a` が常に固定の 8 進値で表現されていますが、この前提がすべてのコードセットで正しいわけではありません。コード内リテラルを使用する代わりに、関数を使用してください。 `getchar()` 関数を使用する次の文では、8 進値の代わりに文字定数を使用しています。

```
if ((c = getchar()) == 'a')
```

ただし、`getchar()` 関数はバイト単位で操作するため、入力ストリーム中の次の文字が複数バイトに渡っているときは正しく動作しません。この問題を回避するには、`getchar()` 関数を `getwchar()` 関数に置き換えます。`getwchar()` 関数は、例で使用されているように、どのコードセットでも正しく動作します。これは、`a` がポータブル文字セットのメンバで、すべてのロケールで同じワイド文字値に変換されるためです。

```
if ((c = getwchar()) == L'a')
```

X/Open の UNIX 標準では、文字定数と文字列リテラル中の、ソース文字セットのすべてのメンバとエスケープ・シーケンスは、すべてのロケールで実行文字セットの同じメンバに変換されると規定しています。そのため、ポータブル文字セット中の文字は、文字定数として、あるいは文字列リテラルで使用しても安全です。英語以外の文字はポータブル文字セットに含まれていないため、リテラルとして使用すると正しく変換されないことがあります。次に例を示します。

```
if ((c = getwchar()) == L' à ')
```

アクセント文字 à は、コードセットのソース文字セットまたは実行文字セットに含まれていない可能性があります。また、アクセント文字のバイナリ値はコードセット間で変換できないこともあります。ソース・ファイルで英語以外の文字が定数として使用されている場合、その結果は定義されていません。このような場合、Unicode ロケールを一貫して使用すると効果があります。

次の例は、何らかの理由で英語以外の文字になる可能性がある定数との比較を行う方法を示します。この定数は、メッセージ・カタログ中のシンボル識別子 MSG\_ID で定義されています。例に示されている文は、MSG\_ID の値をメッセージ・カタログから取り出します。この値は、ロケールに固有で、実行時にプログラムにバインドされます。

```
⋮
char *schar;           ❶
wchar_t wchar;         ❷
⋮

schar = catgets(catd,NL_SETD,MSG_ID,"a"); ❸
if (mbtowc (&wchar,schar,MB_CUR_MAX) == -1) ❹
    error();
if ((c = getwchar()) == wchar) ❺
⋮
```

❶ char を schar へのポインタとして宣言します。

❷ 変数 wchar を wchar\_t 型として宣言します。

❸ catgets() 関数を呼び出して、ユーザのロケールのメッセージ・カタログから MSG\_ID の値を取り出します。

catgets() 関数は値をバイトの配列として返すため、値は schar 変数に返されます。アクセント文字がロケールのコードセットに含まれ

ていない場合、テストはアクセントの付かない基本文字 (a) に対して行われます。

- ④ `schar` に含まれる値が有効なマルチバイト文字を表しているかどうかをテストします。値が有効なマルチバイト文字の場合、プログラムはその文字をワイド文字値に変換し、結果を `wchar` に格納します。

`schar` が有効なマルチバイト文字を含んでいない場合は、プログラムはエラーを発行します。

- ⑤ `wchar` の値を定数として含む条件文です。

メッセージ・カタログと `catgets()` 関数の詳細については、第 3 章を参照してください。マルチバイト文字とマルチバイト文字列を、プログラムで処理可能なワイド文字データに変換する方法については、2.2.4 項を参照してください。

### 2.2.3 マルチバイト文字の操作

Tru64 UNIX は、マルチバイト文字を含むコードセットをサポートするのに必要なすべてのインタフェース (`putwc()`, `getwc()`, `fputws()`, `fgetws()` など) を備えています。マルチバイト文字のサポートを可能にするロケールと機能を組み込むためには、オペレーティング・システムに言語別サブセットをインストールしなければなりません。マルチバイト・ロケールがインストールされていないシステムや、インストールされているが実行時にプログラムにバインドされないシステムでは、`*ws*()` と `*wc*()` 関数は、単に対応するシングルバイト関数 (`putc()`, `getc()`, `fputs()`, `fgets()` など) と同じ動作をします。

### 2.2.4 マルチバイト文字データとワイド文字データ間の変換

国際化に対応しているシステムでは、データはマルチバイト文字とワイド文字のどちらでもエンコードできます。

一般に、マルチバイト・エンコーディングは、データをファイルに格納するとき、あるいはデータを外部用途やデータ交換のために生成するときに使用します。マルチバイト・エンコーディングには、次のような欠点があります。

- 文字は、同じコードセット中でも、1 文字当たりのバイト数が固定した文字として表現されない。そのため、マルチバイト・データ・レコード内の文字サイズは、文字ごとに異なることがあります。

- マルチバイト・データ・レコードから文字コードを読み出すための解析規則はロケールに依存する。

マルチバイト・エンコーディングにはこのような欠点があるため、プログラムの内部処理には、一般に 1 文字当たりのバイト数が固定されているワイド文字エンコーディングが使用されます。実際、ワイド文字形式のデータを内部処理コードともいいます。ワイド文字のサイズはシステムの実装によって異なります。Tru64 UNIX システムでは、ワイド文字のサイズは、HP Alpha プロセッサの性能を最適化する、4 バイト (32 ビット) に設定されています。

テキストのプリントや、スキャン、入出力を行うライブラリ・ルーチンは、操作の種類に適した形でマルチバイト文字からワイド文字へ、あるいはワイド文字からマルチバイト文字へ自動的に変換する機能を備えています。ただし、ほとんどのアプリケーションには、マルチバイト文字への変換やマルチバイト文字からの変換を明示的に指定する文や条件が含まれています。

次の例は、従業員データのデータベースからレコードを読み取るプログラム・モジュールの一部です。この例では、プログラムは固定長のデータを扱いたいので、`mbstowcs()` 関数を使用して、従業員の名前と姓をマルチバイト文字からワイド文字のエンコーディングに変換します。

```
/*
 * The employee record is normalized with the following format, which
 * is locale independent:  Badge number, First Name, Surname,
 * Cost Center, Date of Join in the 'yy/mm/dd' format. Each field is
 * separated by a TAB. The space character is allowed in the First
 * Name and Surname fields.
 */
static const char *dbOutFormat = "%ld\t%S\t%S\tS\t%02d/%02d/%02d\n";
static const char *dbInFormat = "%ld %[^\\t] %[^\\t] %S %02d/%02d/%02d\n";
:
:

    sscanf(record, dbInFormat,
           &emp->badge_num,
           firstname,
           surname,
           emp->cost_center,
           &emp->date_of_join.tm_year,
           &emp->date_of_join.tm_mon,
           &emp->date_of_join.tm_mday);
    (void) mbstowcs(emp->first_name, firstname, FIRSTNAME_MAX+1);
    (void) mbstowcs(emp->surname, surname, SURNAME_MAX+1);
:
:
```

マルチバイト・データを直接処理できる関数のリストについては、A.9 節を参照してください。

## 2.2.5 ソースおよび実行コードセットにおけるマルチバイト文字の規則

同じコードセットのソース文字セットと実行文字セットは、どちらもマルチバイト文字を含むことができます。エンコーディングが同じである必要はありませんが、ソースと実行のどちらの文字セットも、X/Open の標準 UNIX 仕様を満たすコードセットの規則に従わなければなりません。PC コードセットと UCS ベースのコードセットは、これらの規則のいくつか、またはほとんどを満たすだけでかまいませんが、X/Open の UNIX 標準に準拠する UNIX システムに固有のコードセットは、これらの規則をすべて満たさなければなりません。

- ポータブル文字セットに定義されている文字は、両方のセットに含まれていなければならない。
- それ以外のメンバの有無、その意味、およびエンコーディングは、ロケールによって異なる。
- 文字のエンコーディングは状態に依存することがある。文字列にはシフト状態文字が含まれることがあり、シフト状態文字は次のシフト状態文字が検出されるまで、システムにおけるバイトの解釈に影響を与えます。
- 初期シフト状態の間は、基本文字セットのすべての文字は通常どおりに解釈され、シフト状態を変更することはない。
- シーケンス中のバイトの解釈は、現在のシフト状態に依存する。
- すべてのビットがゼロのバイトは、シフト状態とは関係なくヌル文字として解釈される。
- すべてのビットがゼロのバイトは、マルチバイト文字の 2 バイト目以降に現れてはならない。

コードセットのソース・バージョンは、次の規則にも従わなければなりません。

- コメント、文字列リテラル、文字定数、あるいはヘッダ名は、初期シフト状態で始まり、終了しなければならない。
- コメント、文字列リテラル、文字定数、あるいはヘッダ名は、有効なマルチバイト文字の並びで構成されなければならない。

cc コマンド行で `-std1` または `-std` フラグを指定すると、C 言語コンパイラは 3 文字表記 (trigraph sequence) をサポートします。ANSI C 仕様で規定されている 3 文字表記を使用することにより、ソース・コードセットのすべ

ての文字をサポートしていないキーボードからでも，すべての基本文字をプログラムに入力できます。現在定義されている 3 文字表記を以下に示します。各文字表記は，対応する 1 つの文字に置き換えられます。

3 文字表記	対応する文字
??=	#
??(	[
??/	\
??'	^
??<	{
??)	}
??!	
??>	}
??-	~

### 2.2.6 文字の分類

ロケールに依存するプログラムの動作のもう 1 つの特徴に，文字分類が挙げられます。つまり，特定の文字コードがアルファベットの大文字，アルファベットの小文字，数字，句読点，制御文字，あるいはスペース文字のいずれを表しているのか判定する処理のことです。

従来，多くのプログラムは，文字の値が特定の数値範囲に収まっているかどうかを基準にして，文字の分類を行ってきました。たとえば，次の文はすべてのアルファベットの大文字をテストします。

```
if (c >= 'A' && c <= 'Z')
```

この文は，すべての大文字が 0x41 から 0x5a (A から Z) までの範囲に収まっている ASCII コードセットでは有効です。ただし，大文字が 0x41 から 0x5a まで，0xc0 から 0xd6 まで，および 0xd8 から 0xdf までの範囲にある ISO 8859-1 コードセットでは有効ではありません。EBCDIC コードセットでも文字値は異なっており，英語の大文字のエンコーディングそのものが違います。

国際化プログラムを作成するときは，適切な国際化関数を呼び出して文字分類を行います。次に例を示します。

```
if (iswupper (c))
```

国際化関数は、ユーザのロケールの `ctype` 情報に基づいてワイド文字のコード値を分類します。文字分類関数のリストと説明については、A.2 節を参照してください。

### 2.2.7 文字の変換

国際化プログラムで行ってはいけないことの例として、次の文について考えてみます。この文は、`a_var` 内の文字をまず小文字に変換し、次に大文字に変換することによって、ASCII 文字の変換を行います。

```
a_var |= 0x20;  
:  
:  
  
a_var &= 0xdf;
```

ただし、上記の文は、ASCII コードの文字値を前提としていることと、入力不正な値でも変換してしまうことから、国際化プログラムで使用するのには適切ではありません。

大文字と小文字の変換を正しく行うためには、小文字への変換には `towlower()` 関数を、大文字への変換には `toupper()` 関数を呼び出します。

```
a_var = tolower(a_var);  
:  
:  
  
a_var = toupper(a_var);
```

これらの関数は、ユーザのロケールで指定されている情報を使用するため、文字が定義されているコードセットには依存しません。また、これらの関数は、入力が無効であれば引数を変更せずに返します。大文字と小文字の変換関数については、A.3 節を参照してください。

### 2.2.8 文字列の比較

UNIX システムでは、文字列比較のための関数が提供されています。たとえば、次の文は文字列 `s1` と `s2` を比較し、マシンの照合順序で `s1` の値が `s2` よりも大きい、等しい、あるいは小さいかに応じて、それぞれ、ゼロよりも大きい値、等しい値、あるいは小さい値を返します。

```
:  
:  
int cmp_val;  
char *s1;  
char *s2;
```

```

:
cmp_val = strcmp(s1, s2);
:

```

ただしほとんどの言語では、単なる数値的なソートではなく、より複雑な照合アルゴリズムが必要になります。複数回のソート・パスが必要になる理由を次に示します。

- 言語の特定の文字クラスにおけるアクセント文字の順序付けのため (たとえば, a, á, à など)。
- 特定の複数文字のシーケンスを 1 つの文字として照合するため (たとえば, ウェールズ語の文字 ch は, c の後, d の前に並べられます)。
- 特定の 1 文字を 2 文字のシーケンスとして照合するため (たとえば, ドイツ語の文字 sharp s は, ss として照合されます)。
- 照合の際に特定の文字を無視するため (たとえば, 辞書の単語に含まれるハイフンは無視されます)。

国際化環境における文字列比較はコードセットと言語に依存します。そのため、ユーザのロケールの照合順序情報に従って文字列を比較するためには、別の関数が必要になります。そのような関数の例を次に示します。

- `strcoll()`  
この関数は、`strcmp()` のような単純な数値比較ではなく、ユーザのロケールの照合順序情報に従ってワイド文字列を比較します。
- `wscoll()`  
この関数は、`strcoll()` と同じ動作をしますが、ワイド文字列を処理します。
- `wcsxfrm()`  
この関数は、ユーザのロケールの照合順序情報に従ってワイド文字列を変換し、結果として得られた文字列が `wcscmp()` 関数で比較できるようにします。  
  
2 つの文字列が同じかどうかだけを比較するときは、`strcmp()` または `wcscmp()` を使用できます。これらの関数は、ほとんどの環境で `wscoll()` よりも高速に実行します。



## 2.3 文化的データの処理

文化的データとは、言語や地域によって異なる可能性がある情報項目のことです。

次に例を示します。

- 英国と米国では、ピリオドは小数点を表し、コンマは 10 進数の千単位の区切り文字を表すが、ドイツでは 10 進数でのこの 2 つの文字が逆の意味に使用される。
- 米国では 1986 年 10 月 7 日を 10/7/1986 と表記するが、英国では同じ日付を 7/10/1986 と表記する。この例は、同じ言語を使用しているも、文化的データ項目が異なることがあることを示します。
- 日付の区切り文字と、年、月、および日の順序は、国によって異なる。たとえばドイツでは、1986 年 10 月 7 日は 7/10/86 ではなく、7.10.86 と表記されます。
- 通貨記号は、使用される文字と、それが金額値のどこに置かれるかという 2 点で異なる。通貨記号は値の前、後、あるいは値の中に置かれることがあります。

欧州経済通貨連合に所属する国で通貨記号として使用されるユーロ文字は、Unicode (\*.UTF-8) または Latin-9 (\*.ISO8859-15) ロケールと、対応するフォントでのみサポートされています。ユーロ通貨記号のサポートについての詳細は、euro(5) を参照してください。

キーボードからユーロ文字を入力するには、Latin-9 または UTF-8 のロケールで作業し、適切なキーマップがアクティブでなければなりません。ユーロ文字を表示するには、Latin-9 または UTF-8 のロケールで作業し、適切なフォントがアクティブでなければなりません。必要なロケールと、適切なキーマップおよびフォントをアクティブにするには、Latin-9 または UTF-8 のロケールの環境にログインするか、setenv を使って LANG 環境変数を設定し、新たに dtterm を起動します。リファレンス・ページ locale(1) および dtterm(1) を参照してください。

国際化プログラムを作成するときは、文化的データについて前提を設けることはできません。プログラムは、ユーザの地域固有の習慣に従って動作しなければなりません。X/Open の UNIX 標準では、プログラムが実行時にアクセスできる文化的データ項目のデータベースと、それに関連するインタフェース群を利用して、この要件が満たされるように規定しています。これ

以降の項では、このデータベースと、そのデータ項目の抽出と処理に使用される関数について説明します。

### 2.3.1 langinfo データベース

langinfo と呼ばれる言語情報データベースには、システム上でサポートされている各ロケールの文化情報の詳細を表す項目が含まれています。langinfo データベースは、X/Open の UNIX 標準の規定に従い、ロケールごとに次の情報を含みます。

- コードセット名
- 日付と時刻の書式
- 曜日の名前
- 月の名前
- 曜日の省略形
- 月の省略形
- 小数点 (整数と小数部を区切る文字)
- 千単位の区切り文字
- yes/no の問い合わせに対する肯定と否定の応答
- 通貨記号と、金額値中での通貨記号の位置
- 元号の名前と年 (日本語のロケール用)

### 2.3.2 langinfo データベースの照会

`nl_langinfo()` 関数を呼び出すことにより、langinfo データベースから文化的データ項目を抽出できます。この関数は、ヘッダ・ファイル `/usr/include/langinfo.h` に定義されている定数の 1 つを *item* 引数として取ります。この関数は、現在のロケールにおける *item* の値を含む文字列へのポインタを返します。

次の例は、日付と時刻情報を書式付けるための文字列を抽出する `nl_langinfo()` の呼び出しです。この値は、定数 `D_T_FMT` に対応します。

```
nl_langinfo(D_T_FMT);
```

### 2.3.3 ローカルな習慣に合った日付および時刻文字列の生成と解釈

プログラムでは日付と時刻の文字列が生成されることがよくあります。国際化プログラムはユーザのローカルな習慣に合った文字列を生成します。この場合、要件を満たすには、`strftime()` または `wcsftime()` 関数を呼び出します。どちらの関数も `langinfo` データベースを間接的に使用します。さらに、`wcsftime()` 関数は、日付と時刻をワイド文字形式に変換します。

次の例では、`strftime()` 関数は、`langinfo` データベースの `D_FMT` 項目で定義されている日付文字列を生成します。

```
⋮
setlocale(LC_ALL, ""); ❶
⋮

clock = time((time_t*)NULL); ❷
tm = localtime(&clock); ❸
⋮

strftime(buf, size, "%x", tm); ❹
puts(buf); ❺
⋮
```

- ❶ 実行時にプログラムを、システムまたは個々のユーザのロケール・セットにバインドします。
- ❷ `time()` サブルーチンを呼び出して、時刻値を `clock` 変数に返します。返される時刻値は、協定世界時からの相対値です。
- ❸ `localtime()` 関数を呼び出して、`clock` に格納されている値を、`tm` 構造体に格納できる値に変換します。`tm` 構造体は、年、月、日、時間、分などの値を表すメンバを含みます。
- ❹ `strftime()` 関数を呼び出して、`tm` 構造体に格納されている値から、ユーザのロケールで定義されている書式の日付文字列を生成します。

`buf` 引数は、日付文字列が返される文字列変数へのポインタです。  
`size` 引数は、`buf` の最大サイズを含みます。`"%x"` 引数は、`printf()` と `scanf()` 関数で使用する書式文字列と同様な変換指定を指定します。`"%x"` 引数は出力文字列中で、ロケールでの適切な表現に置き換えられます。

- ⑤ `puts()` 関数を呼び出して、`buf` に含まれている文字列を標準出力ストリーム (`stdout`) にコピーし、改行文字を付加します。

`strftime()` と `nl_langinfo()` を組み合わせて、日付と時刻の文字列を生成する方法を示す、次の例を考えてみます。前の例と同様に、`setlocale()`、`time()`、および `localtime()` インタフェースの呼び出しが、ここでも行われているものとします。ただし、次の例では、`strftime()` の呼び出しで、書式文字列引数が `nl_langinfo()` の呼び出しに置き換えられています。

```
⋮
⋮
strftime(buf, size, nl_langinfo(D_T_FMT), tm);
puts(buf);
⋮
⋮
```

文字列から日付と時刻値への変換 (つまり、`strftime()` の逆の動作) には、`strptime()` 関数を使用します。`strptime()` 関数では多数の変換指定子がサポートされていますが、その動作はロケールに依存します。

### 2.3.4 金額値の書式付け

`strfmon()` 関数は、実行時にプログラムにバインドされるロケール情報に従って、金額値を書式付けます。次に例を示します。

```
strfmon(buf, size, "%n", value);
```

この文は、`value` 変数に含まれている倍精度浮動小数点値を書式付けます。`"%n"` 引数は、実行時のロケールが定義する書式に置き換えられる書式指定です。結果は `buf` 配列に返されます。この配列の最大長は、`size` 変数に格納されています。

`money` プログラムは、`strfmon()` 関数がどのように動作するかを示します。ワールドワイド言語サポート・サブセットをインストールする場合、このサンプル・プログラムのソース・ファイルは `/usr/i18n/examples/money` ディレクトリにインストールされます。

### 2.3.5 プログラム独自の形式による数値の書式付け

数値、金額値などの独自の変換を行う場合は、ユーザのロケール内の特定の書式情報を使用できます。`localeconv()` 関数は引数を取らない関数であ

り、ロケールで定義されている数値の書式付けに関するすべての情報を、プログラム中で宣言されている構造体に返します。次に例を示します。

```
struct lconv *app_conv;
```

プログラムで定義されているルーチンは、`lconv` 構造体に格納されている次の情報を使用できます。

- 小数点文字
- 千単位の区切り文字
- 桁のグループ・サイズ
- 国際通貨記号
- 国内通貨記号
- 金額値の小数点文字
- 金額値の千単位の区切り文字
- 金額値の桁のグループ・サイズ
- 正符号
- 負符号
- 表示される小数点以下の桁数
- 負の金額値に使用するカッコ記号

### 2.3.6 他の処理における `langinfo` データベースの使用

これまでに説明した関数以外にも、`langinfo` データベースを使用して、特定の文化的データ項目の設定を判定する関数があります。たとえば、`wscanf()`、`wprintf()`、および `wcstod()` 関数は、`langinfo` データベースの情報から適切な小数点文字を決定します。

## 2.4 テキストの表示と入力の処理

アプリケーションを作成する場合、次の3つの事柄に関して、ユーザの言語を考慮しなければなりません。

- プログラム・メッセージの定義とアクセス方法 (2.4.1 項)
- プログラムにおける出力テキストの表示方法 (2.4.2 項)
- プログラムにおける入力テキストの処理方法 (2.4.3 項)

## 2.4.1 メッセージの作成と使用

プログラムは、ユーザが使用する言語でユーザと対話しなければなりません。そのために、プログラム・メッセージをどのように定義して、またどのようにアクセスするかについては、いくつかの制約が課せられています。具体的には、メッセージはプログラム・ソース・コードとは別のファイルで定義されており、オブジェクト・ファイルにコンパイルされることはありません。メッセージは別のファイルに含まれるため、メッセージを別の言語に翻訳し、実行時にプログラムにリンクされる形態で格納できます。このようにしておけば、プログラムは、ユーザの言語に合わせて翻訳されたメッセージ・テキストを取り出すことができます。

X/Open UNIX 標準では、次のメッセージ関数が規定されています。

- メッセージ・テキスト・ソース・ファイルの定義を含むメッセージ・システム
- ソース・ファイルからメッセージ・カタログを生成する `gencat` コマンド
- 1 つまたは複数のカタログから個々のメッセージを実行時に取り出すライブラリ関数群

次に、国際化プログラムがカタログからメッセージを取り出す例を示します。

```
#include <stdio.h> [1]

#include <locale.h> [2]
#include <nl_types.h> [3]
#include "prog_msg.h" [4]
main()
{
    nl_catd catd; [5]
    setlocale(LC_ALL, ""); [6]
    catd = catopen("prog.cat", NL_CAT_LOCALE); [7]
    puts(catgets(catd, SETN, HELLO_MSG, "Hello, world!")); [8]
    catclose(catd); [9]
}
:
```

- [1] 標準 C ライブラリ用のヘッダ・ファイルをインクルードします。
- [2] `setlocale()` 関数と、関連する定数および変数を宣言するヘッダ・ファイル `/usr/include/locale.h` をインクルードします。

- ③ `catopen()` , `catgets()` , および `catclose()` 関数を宣言するヘッダ・ファイル `/usr/include/nl_types.h` をインクルードします。

- ④ プログラム固有のヘッダ・ファイル `prog_msg.h` をインクルードします。このヘッダ・ファイルには、このプログラム・モジュールが使用するメッセージ・セット (SETN) と、個々のメッセージ (この例の `HELLO_MSG`) を識別する定数が設定されています。

メッセージ・カタログには、1 つまたは複数のメッセージ・セットを含めることができます。各セットには、個々のメッセージが配置されています。

- ⑤ メッセージ・カタログ記述子 `catd` を、`nl_catd` 型として宣言します。この記述子は、カタログをオープンする関数によって返されます。またこの記述子は、カタログをクローズする関数への引数としても渡されます。

- ⑥ `setlocale()` 関数を呼び出して、プログラムのロケール・カテゴリを、ユーザのロケール環境変数の設定にバインドします。

`LC_MESSAGES` カテゴリに設定されているロケール名は、この例の `catopen()` と `catgets()` 関数によって使用されるロケールです。一般に、システム管理者やユーザは、`LANG` または `LC_ALL` 環境変数だけを特定のロケール名に設定します。これにより、`LC_MESSAGES` 変数も暗黙的に設定されます。

- ⑦ `catopen()` 関数を呼び出して、このプログラムが使用する `prog.cat` というメッセージ・カタログをオープンします。

`NL_CAT_LOCALE` 引数は、`LC_MESSAGES` に設定されているロケール名をプログラムで使用することを指定します。`catopen()` 関数は、`NLSPATH` 環境変数に設定されている値を使用して、メッセージ・カタログの位置を判別します。この呼び出しは、`catd` 変数にメッセージ・カタログ記述子を返します。

- ⑧ `puts()` 関数を呼び出して、メッセージを表示します。

この呼び出しの第 1 引数は、`catgets()` 関数の呼び出しです。この関数は、`HELLO_MSG` 識別子を持つメッセージのテキストを取り出します。このメッセージは、`SETN` 定数によって識別されるメッセージ・セットに含まれています。`catgets()` の最後の引数は、メッセージ呼び出しがカタログから翻訳テキストを取り出せなかった場合に使用される、省略時のテキストです。一般に、省略時のテキストは英語です。

- ⑨ `catclose()` 関数を呼び出して、記述子が `catd` 変数に格納されているメッセージ・カタログをクローズします。

メッセージ・カタログの作成と使用については、第3章を参照してください。

## 2.4.2 出力テキストの書式付け

メッセージをさまざまな言語に翻訳するためには、メッセージをプログラム・ソースから分離するだけでなく、プログラム中のメッセージ文字列を注意深く作成しなければなりません。

次の例を考えてみます。

```
printf(catgets(catd, set_id, WRONG_OWNER_MSG,
               "%s is owned by %s\n"),
       folder_name, user_name);
```

上記の文はメッセージ・カタログを使用していますが、特定の言語構造(名詞の後に受動態の動詞があり、その後に名詞が続く)を想定しています。動詞の受動態は、すべての言語にあるわけではありません。そのため、翻訳されたメッセージでは、`folder_name` の前に `user_name` がプリントされる可能性があります。つまり、表示されるメッセージが、“JULY\_REVENUE is owned by John\_Smith.”ではなく、“John\_Smith owns JULY\_REVENUE”となるように、翻訳者がメッセージの語順を変更しなければならないことがあります。

メッセージ要素の順序が固定されていることから生じる問題を回避するために、`printf()` ルーチンの書式指定子では、次の未使用の引数に対して書式変換を行うだけでなく、引数リスト内の *n* 番目の引数に対して書式変換を行えるようになりました。拡張された書式変換を利用するには、変換文字の `%` を、`%digit$` というシーケンスに置き換えます。*digit* は、引数リスト中での引数の位置を指定します。この機能を書式文字列 “%s is owned by %s\n” に適用した例を、次に示します。

```
printf(catgets(catd, set_id, WRONG_OWNER_MSG,
               "%1$s is owned by %2$s\n"),
       folder_name, user_name);
```

翻訳者は、プログラムが使用するメッセージ・ファイル内の `WRONG_OWNER_MSG` エントリの省略値、つまり、文字列 “%1\$s is owned by %2\$s” を、次の語順を持つ英語以外の言語に置き換えることができます。

```
WRONG_OWNER_MSG      "%2$s owns %1$s\n"
```



### 2.4.3 入力テキストの走査

2.4.2 項で説明した、出力テキストの書式付けに関する問題は、入力テキストでも起こります。たとえば、ユーザが日付の要素を指定する順序や、金額文字列の各部分を区切るために入力する文字は、国によって異なります。そのため、`scanf()` ファミリの関数でも、ユーザが文字列の要素を入力する方法に柔軟性を持たせるために、拡張書式変換指定子がサポートされています。

次の例について考えてみます。

```
⋮
int day;
int month;
int year;
⋮

scanf("%d/%d/%d", &month, &day, &year);
⋮
```

この文の書式文字列は、すべてのユーザが日付の入力に米国英語の書式 (月/日/年) を使用するという前提に基づいています。国際化プログラムでは、拡張書式指定子を使用して、文字列要素の順序に関する言語固有の要件をサポートするようにします。次に例を示します。

```
⋮
scanf(catgets(catd, NL_SETD, DATE_STRING,
               "%1$d/%2$d/%3$d"), &month, &day, &year);
⋮
```

DATE\_STRING メッセージの省略値である "%1\$d/%2\$d/%3\$d" は、ユーザが日付の入力に月/日/年の書式を使用する国でのみ有効です。文字列要素の順序や書式が異なる国では、翻訳者はプログラムのメッセージ・ファイルのエントリを変更できます。次の言語について考えてみます。

- イギリス英語 (日/月/年)

```
DATE_STRING      "%2$d/%1$d/%3$d"
```

- ドイツ語 (日.月.年)

```
DATE_STRING      "%2$d.%1$d.%3$d"
```

## 2.5 実行時環境へのロケールのバインド

国際化プログラムが正しく動作するためには、ユーザの環境に適した地域化されたデータに実行時にバインドしなければなりません。この作業は `setlocale()` 関数が行います。 `setlocale()` を呼び出すと、次の処理が実行されます。

- ユーザのプロセスですでに有効になっているロケール設定にバインドする。
- プログラムが制御するロケール設定にバインドする。
- ロケール設定を変更せずに、現在のロケール設定を照会する。

この呼び出しは、`category` と `locale_name` の 2 つの引数を取ります。

`category` 引数は、照会したい、変更したい、あるいは使用したいロケールのすべての、または特定のセクションを指定します。 `category` の値と、その意味を次に示します。

- `LC_ALL`

この `category` 引数は、ロケールのすべてのセクションを指定します (各セクションの指定より優先されます)。

- `LC_CTYPE`

この `category` 引数は、大文字小文字変換などの操作で使用するクラスと文字属性を定義します。

- `LC_COLLATE`

この `category` 引数は、ソートや照合の操作での文字や文字列の順序を指定します。

- `LC_MESSAGES`

この `category` 引数は、肯定および否定応答と、プログラム・メッセージを指定します。

- `LC_MONETARY`

この `category` 引数は、金額値に使用される規則や特殊記号を指定します。

- `LC_NUMERIC`

この `category` 引数は、数値の書式付けに使用される規則や特殊記号を指定します。

- `LC_TIME`

この *category* 引数は、曜日や月の名前と省略形、日付と時刻の表記を決定するその他の文字列と書式変換を指定します。

*locale\_name* 引数は、次の値のいずれかです。

- 空の文字列 (""). 実行時にシステム管理者またはユーザが、*category* に設定されたロケール名にプログラムをバインドします。
- ロケール名。 *category* にすでに設定されているロケールを変更します。
- `NULL`。 *category* に現在設定されているロケール名を調べます。

### 2.5.1 システムまたはユーザのロケール・セットへのバインド

一般に、システム管理者やユーザは、`LANG` または `LC_ALL` 環境変数にロケール名を設定します。これらの環境変数のいずれかを設定すると、ロケールのすべてのロケール・カテゴリ変数に同じロケール名が自動的に設定されます。

`LC_ALL` を使用してすべてのロケール・カテゴリに同じロケール名を設定した場合以外は、システム管理者やユーザは、個々のロケール・カテゴリ変数に異なるロケール名を設定できます。一般に国際化プログラムには、プログラム内のすべてのロケール・カテゴリを、そのユーザに対してすでに有効になっている環境変数設定で初期化するための、`LC_ALL` 呼び出しが含まれています。例を次に示します。

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

標準のロケール名は、`language_TERRITORY.codeset@modifier` からなります。たとえば、`zh_CN.dechanzi@radical` です。意味は、次のとおりです。

- `language`。ロケールの言語を示します (zh は中国語)。
- `_TERRITORY`。ロケールの地理的な国や領域 (\_CN は中国、これに対して TW は台湾、HK は香港)。
- `.codeset`。ロケールで使用するコード化文字セット (dechanzi)。
- `@modifier`。ロケールの地域化データの追加情報 (radical による照合)

ロケールには通常、複数のバリエーションがあります。各バリエーションの名前はベースのロケールと同じですが、アット記号 (@) で始まるファイル名サフィックスが付いています。本来 UNIX にはないコードセット (UCS-4

や CP850 など) をサポートするロケール・バリエーションは、LANG や LC\_ALL に設定できます。

ただし、1 つのロケール・カテゴリだけベース・ロケールと異なっているロケール・バリエーションは、適切なロケール・カテゴリにのみ割り当ててください。たとえば、@radical などの特定の照合シーケンスをサポートするためのロケール・バリエーションは、LC\_COLLATE に設定します。ユーロ通貨記号 (@euro) をサポートするためのロケール・バリエーションは、LC\_MONETARY に設定します。LANG 環境変数の設定には、これらのバリエーションではなく、ベース・ロケール名を使用してください。

さらに、ベース・ロケール名をすべてのロケール・カテゴリに設定するの でなければ、LC\_ALL 環境変数は使用しないでください。この環境変数に値を設定すると、LANG と、個別のロケール・カテゴリの環境変数の設定が無効になります。

多くのロケール固有ファイルは、ロケール名の言語、地域、およびコードセット部分からなる名前のディレクトリ内に置かれます。コマンドやその他のシステム・アプリケーションは、ディレクトリ・ノードの 1 つとして %L を含む検索パスに、LANG 変数の設定値を挿入します。これにより、ソフトウェア・プログラムは、現在のロケールで使用されるべき、フォント、リソース・ファイル、ユーザ定義文字ファイル、翻訳済みのリファレンス・ページなどの、正しいセットを見つけることができます。LANG 変数に設定する値に、照合に関連する @ サフィックスが含まれていると、ロケール固有のファイルの一部がアプリケーションから見つからなくなることがあります。

## 2.5.2 プログラム実行時のロケールの変更

国際化プログラムの中には、ユーザからのロケール名の入力を受け付けたり、プログラムの実行中にロケールを変更しなければならないものがあります。次の例は、setlocale() を呼び出して、すべてのロケール・カテゴリを同じロケール名で明示的に初期化または再初期化します。

```
⋮
nl_catd catd; [1]
char buf[BUFSIZ]; [2]
⋮

setlocale(LC_ALL, ""); [3]
catd = catopen(CAT_NAME, NL_CAT_LOCALE); [4]
⋮
```

```
printf(catgets(catd, NL_SETD, LOCALE_PROMPT_MSG,
               "Enter locale name: ")); 5
gets(buf); 6
setlocale(LC_ALL, buf); 7
:
```

1 カタログ記述子 `catd` を `nl_catd` 型として宣言します。

2 後でロケール名が格納される `buf` 変数を宣言します。

プログラムを実行するシステムに関係なく、この変数がロケール名を格納できる十分な大きさになるように、最大サイズを `BUFSIZ` に設定します。この定数は、システム・ベンダが `/usr/include/stdio.h` で定義しています。

3 `setlocale()` を呼び出して、プログラムのロケール設定を、プログラムを実行するユーザに対して有効になっているロケール設定に初期化します。

4 `catopen()` を呼び出して、プログラムのメッセージが格納されているメッセージ・カタログをオープンします。この関数は、カタログ記述子を `catd` 変数に返します。

`CAT_NAME` 定数は、プログラム自体のヘッダ・ファイルで定義されています。

5 ユーザに新しいロケール名の入力を求めます。

`NL_SETD` 定数は、メッセージ・カタログ中の省略時のメッセージ・セット番号を指定するもので、`/usr/include/nl_types.h` で定義されています。`LOCALE_PROMPT_MSG` 識別子は、省略時のメッセージ・セット内の翻訳されたプロンプト文字列を指定します。

6 `gets()` 関数を呼び出して、ユーザが入力したロケール名を読み取り、`buf` 変数に格納します。

7 `locale_name` 引数に `buf` を指定して `setlocale()` を呼び出し、ロケール全体を再初期化します。

場合によっては、特定のカテゴリのデータについてのみロケールを変更しなければならないことがあります。たとえば、金額値を含んでいる、いくつかの国固有のファイルを処理するプログラムは、各ファイルのデータを処理す

る前に、プログラム変数を新しいロケール名で再初期化し、その変数値を使用してロケールの `LC_MONETARY` カテゴリだけを再設定できます。

---

## メッセージ・カタログの作成と使用

メッセージ・カタログは、プログラムからアクセスできる地域化データのファイルです。langinfo データベースと同じ定義が適用されますが、両者には相違点もあります。

langinfo データベースの地域化データ要素は、オペレーティング・システムによって提供されるライブラリ・ルーチン、コマンド、およびユーティリティを含むすべてのアプリケーションで使用されます。langinfo データベースは、ロケールを定義するソース・ファイルから作成されます。

langinfo データベースとは異なり、メッセージ・カタログは、1つのプログラム、または関連するプログラム・セットの地域化に必要な情報を提供します。メッセージ・カタログは、メッセージ・テキストのソース・ファイルから生成されます。このファイルには、言語や文化の違いによって変更しなければならない、エラーや情報メッセージ、プロンプト、フォームのバックグラウンド・テキスト、さまざまな文字列や定数が含まれます。

グラフィカル・ユーザ・インタフェースを使用する X アプリケーションや Motif アプリケーションは、タイトル・バー、メニュー、ボタン、およびウィンドウ内の簡単なメッセージなどの少量のテキスト・データを処理するときは、これらのデータをメッセージ・カタログではなく、X リソース・ファイルから取り出します。Motif アプリケーションでは、ヘルプやエラー・メッセージなどのテキストにアクセスする際に、テキスト・ライブラリ・ファイルに加え、ユーザ・インタフェース言語 (UIL) ファイルをオプションとして使用できます。ただし、X アプリケーションと Motif アプリケーションはどちらも、メッセージ・カタログ内のテキストにアクセスできます。

この章ではメッセージ・カタログに焦点を当て、以下の各項目について説明します。

- 3.1.1 項。メッセージ・テキスト・ソース・ファイルの内容を記述する際に適用できる、一般的なガイドラインについて説明します。

- 3.1.2 項。メッセージをグループ化するために使用できる、メッセージ・テキスト・ソース・ファイルのオプションの構成要素、メッセージ・セットについて説明します。
- 3.1.3 項。メッセージ・テキスト・ソース・ファイルを構成するメッセージ・エントリについて説明します。
- 3.1.4 項では、quote ディレクティブについて説明し、3.1.5 項では、テキストを区切ったり、実行に影響しないコメントをメッセージ・テキスト・ソース・ファイル内に入力するために使用する、コメント行について説明します。
- 3.1.6 項。メッセージ・テキストの作成時に使用するスタイル・ガイドラインについて説明します。
- 3.2 節。既存のプログラムからメッセージ・テキストを抽出する方法について説明します。
- 3.3 節。メッセージ・テキスト・ソース・ファイルの編集と翻訳方法について説明します。
- 3.4 節。メッセージ・カタログの作成方法 (mkcatdefs コマンドと gencat コマンドの使用法を含む) と、メッセージ・カタログの設計および維持のヒントについて説明します。
- 3.5 節。メッセージとロケール・データの対話型での表示方法と、スク립トからの表示方法について説明します。
- 3.6 節。プログラムからメッセージ・カタログにアクセスする方法 (catopen(), catclose(), および catgets() 関数によるメッセージ・カタログのオープン、クローズ、および読み取りを含む) について説明します。

テキストの取り出しおよび表示に使用するメソッドに関係なく、メッセージ・カタログ・テキストの翻訳に適用される X および Motif プログラミング・ガイドラインについては、3.1.6 項を参照してください。

## 3.1 メッセージ・テキスト・ソース・ファイルの作成

メッセージ・カタログを作成し使用する前に、まずメッセージ・テキスト・ソース・ファイルの構成要素、構文、および意味について理解しておかなければなりません。この章の以降の部分ではソース・ファイルの内容と処理方法について説明しますが、その前に、理解を早めるためにソース・ファイル



の例を簡単に説明します。例 3-1 は、オンラインのサンプル・プログラム xpg4demo のメッセージ・テキスト・ソース・ファイルからの抜粋です。

### 例 3-1: メッセージ・テキスト・ソース・ファイル

```
$ /* 1
$ * XPG4 demo program message catalogue. 1
$ * 1
$ */ 1
2
$quote " 3
$set MSGError 4
E_COM_EXISTBADGE      "Employee entry for badge number %ld \ 5
already exists"
E_COM_FINDBADGE      "Cannot find badge number %ld" 5
E_COM_INPUT          "Cannot input" 5
E_COM_MODIFY          "Data file contains no records to modify" 5
E_COM_NOENT          "Data file contains no records to display" 5
E_COM_NOTDEL         "Data file contains no records to delete" 5
:
:

$set MSGInfo 4
I_COM_NEWEMP          "New employee" 5
I_COM_YN_DELETE       "Do you want to delete this record?" 5
I_COM_YN_MODIFY       "Do you want to modify this record?" 5
I_COM_YN_REPLACE      "Are these the changes you want to make?" 5
:
:

$ NOTE - Message contains the format used to display numeric dates
$ The first descriptor, 1$, contains the year
$ The second descriptor, 2$, contains the month
$ The third descriptor, 3$, contains the day
I_SCR_IN_DATE_FMT     "%2$d/%3$d/%1$d" 6
$set MSGString 4
$
$ One-character commands.
$ Note: These should not be translated because they are keywords for the application.
S_COM_CREATE          "c" 7
S_COM_DELETE          "d" 7
S_COM_EXIT            "e" 7
:
:

$ Note: These are column heads and spacing and should be maintained
$ Column one begins at space 1.
$ Column two begins at space 15.
$ Column three begins on space 37.
$ Column four (an abbreviation of Department) begins at space 60.
$ Column five (an abbreviation of Date of Birth) begins at space 68.
$ S_COM_LIST_TITLE is output to underscore headers and should be
$ increased or decreased as appropriate for translation.
S_COM_LIST_TITLE      "Badge      Name      Surname \
Dept      DOB\n" 8
S_COM_LIST_LINE       "-----\n" 8
:
:
$
$ If surname comes before first name, "y" should be specified.
$
```

### 例 3-1: メッセージ・テキスト・ソース・ファイル (続き)

---

```
S_SCR_SNAME1ST      "n"  9
:
:
```

---

- ❶ ドル記号 (\$) で始まり、その後にスペースまたはタブ文字が続く行は、コメント行です。コメント行については、3.1.5 項で説明します。
- ❷ 読みやすくするために、ファイル内の任意の位置に空白行を入れます。
- ❸ クォート文字は、メッセージ・テキストを区切ります。quote ディレクティブについては、3.1.4 項で説明します。
- ❹ 識別子は、メッセージ・セットの始まりを示します。このソース・ファイルには、エラー・メッセージ (MSGError セットに含まれる)、情報メッセージ (MSGInfo セットに含まれる)、およびその他の文字列と書式 (MSGString セットに含まれる) の、3 つのメッセージ・セットがあります。メッセージ・セットの定義と削除については、3.1.2 項を参照してください。
- ❺ ソース・ファイル内の大半の行は、メッセージ・エントリです。このエントリは、重複しない識別子とメッセージ・テキスト文字列から構成されます。1 番目のメッセージ・エントリは、バックスラッシュ (\) を使用して次の行に継続しています。他にも、\n (改行) のように、メッセージの出力を制御する特別な文字シーケンスを含むエントリがあります。メッセージ・エントリの詳細については、3.1.3 項を参照してください。メッセージ・エントリに適用される規則とオプションについては、3.1.1 項も参照してください。
- ❻ このタイプのメッセージ・エントリは、ユーザにデータ入力を求める際のデータ要素の入力順序を、翻訳者が変更できるようにしています。プログラム・ロジックでメッセージの書式付けを行う方が好ましいですが、メッセージ・エントリを使用して書式を制御することもできます。この行は、変数を識別する必要がある翻訳者に対するコメントも示しています。

- ⑦ このタイプのメッセージ・エントリは、単語の省略形を定義します。別の言語を使用した場合でも、略語が一意になるように配慮する必要があります。
- ⑧ このタイプのメッセージ・エントリは、メニュー表示のためのヘッダ行を定義します。これにより、翻訳者はフィールドの順序や行の長さを、文化習慣の違いを吸収するためにプログラムに用意されている他の調整項目に合わせて調整できます。この行は、略語に慣れていない翻訳者や、書式付けされた欄の幅を知る必要がある翻訳者に対するコメントも示しています。
- ⑨ このタイプのメッセージ・エントリは、プログラムで名前フィールドの位置関係を決める方法を制御する定数を定義します。たとえば、`xpg4demo` プログラムでは、名前と姓の順番を変更できます。

1 つまたは複数のメッセージ・テキスト・ソース・ファイルを使用して、プログラムが実行時にアクセスできるメッセージ・カタログ (`.cat` ファイル) を作成できます。例 3-1 のソース・ファイルからメッセージ・カタログを作成する手順は、次のとおりです。

1. `mkcatdefs` コマンドを使用して、メッセージ・セットとメッセージのシンボリック識別子を、カタログ内でのメッセージ・セットの位置と、各セット内でのメッセージの位置を示す番号に変換します。
2. `gencat` コマンドを使用して、`mkcatdefs` の出力からメッセージ・カタログを作成します。

`mkcatdefs` コマンドと `gencat` コマンドについては、3.4 節で説明します。

### 3.1.1 一般規則

この項では、メッセージ・テキスト・ソース・ファイルの構文に適用される一般的なガイドラインについて説明します。メッセージ・テキストの内容のスタイル・ガイドラインについては、3.1.6 項で説明します。

メッセージ・テキスト・ソース・ファイル (`.msg` ファイル) には、メッセージの並びが入っています。オプションとして、1 つまたは複数のメッセージ・セット内でこれらのメッセージの順序を指定できます。一般にアプリケーションには、地域化ごとに個別のメッセージ・ソース・ファイルが用意されます。つまり、ユーザがアプリケーションを実行するロケールごと

に (コードセット, 言語, および地域の組み合わせごとに) ソース・ファイルがあります。

識別子の値をクオート文字で囲まない場合は, ソース・コードセットで定義されているスペースまたはタブ文字を 1 つ指定して, ソース・ファイルの行中のフィールドを区切ります。スペースやタブを 2 つ以上指定した場合, 余分なスペースやタブは値の一部として扱われます。quote ディレクティブで指定した文字を使用してメッセージ文字列全体を囲むと, 識別子と文字列の間にある余分なスペースやタブが, 文字列の一部として扱われるのを防ぐことができます (quote ディレクティブについては, 3.1.4 項を参照してください)。また, 末尾にスペースやタブを含むメッセージ・テキストでは, メッセージ文字列をクオート文字で囲まなければ, テキストの末尾にスペースやタブが含まれていることを示すことはできません。

メッセージ・テキスト文字列には, 通常の文字に加え, 表 3-1 に示す特殊文字のシーケンスを含めることができます。

表 3-1: メッセージ・テキスト・ソース・ファイルにおける特殊文字の記述

説明	シンボル	文字シーケンス
改行	NL (LF)	\n
水平タブ	HT	\t
垂直タブ	VT	\v
後退	BS	\b
復帰	CR	\r
改ページ	FF	\f
バックスラッシュ	\	\\
8 進数値	ddd	\ddd <sup>a</sup>
10 進数値	dddd	\xdddd <sup>b</sup>

<sup>a</sup>\ddd 形式のエスケープ・シーケンスは, バックスラッシュの後に, 希望の文字の値を指定する 1 ~ 3 個の 8 進数字を続けたものです。

<sup>b</sup>\xdddd 形式のエスケープ・シーケンスは, バックスラッシュの後に, 文字 x と, 希望の文字の値を指定する 1 ~ 4 個の 16 進数字を続けたものです。16 進数の文字シーケンスは X/Open UNIX CAE Specifications の拡張であり, この仕様に準拠する他のシステムではサポートされていないことがあります。

メッセージ・ファイル内のバックスラッシュは, 表 3-1 に記載されていない文字シーケンスが後に続く場合は無視されます。たとえば, \m というシーケンスは, メッセージ中に m を出力します。8 進数値や 16 進数値の形式

で文字を表現する場合、特殊文字の数値表現に続く文字も有効な 8 進数字または 16 進数字であれば、数字の前にゼロを付加します。たとえば、ドル記号の 8 進数が 44 の場合に \$5.00 を出力するには、\0445.00 と指定しなければなりません。この指定により、5 が 8 進数値の一部として解釈されるのを防ぐことができます。

通常、改行文字はメッセージ・エントリを区切ります。ただし、改行文字の直前にバックスラッシュを入力することにより、メッセージ文字列を次の行に継続させることができます。この場合、改行文字の入力では、英語キーボードの Return キーまたは Enter キーを押します。たとえば、次の 2 つのエントリは等価であり、プログラム・ユーザからの見え方は同じです。

```
MSG_ID          This line continues \  
to the next line.  
MSG_ID          This line continues to the next line.
```

メッセージ・ソース・ファイル内の空行は、すべて無視されます。ファイルを読みやすくするために、空行を入力できます。

### 3.1.2 メッセージ・セット

メッセージ・セットは、メッセージ・テキスト・ソース・ファイル内のオプションの構成要素であり、省略可能です。メッセージ・セットは、使用目的を問わず、メッセージをグループ化するのに使用できます。複数のプログラム・ソース・ファイルから構築されるアプリケーションでは、メッセージ・セットを作成し、プログラム・モジュールに対応させてメッセージを構成したり、あるいは、オンラインのサンプル・プログラム xpg4demo のように、同じカテゴリ (エラー、情報、定義文字列) に属するメッセージをグループ化することができます。

プログラム・モジュールごとにメッセージをグループ化する利点は、後でアプリケーションからモジュールを削除するときに、そのモジュールに対応するメッセージを簡単に見つけてカタログから削除できることです。

メッセージを使用目的別にグループ化すると、同じアプリケーション内の複数のモジュール間でメッセージを共有できます。使用目的別にグループ化することにより、アプリケーションの新しいモジュールを作成したり、既存のモジュールを保守するプログラマは、必要なメッセージがすでにファイル内に存在するかどうかを簡単に調べることができます。

set ディレクティブは、後続のメッセージのセット識別子を指定し、次の set ディレクティブまたはファイルの終わりが検出されるまでのメッセージをそのセットに含めます。set ディレクティブの形式は次のとおりです。

```
$SET set_id [ comment]
```

set\_id 変数は、次のいずれかです。

- [1 - NL\_SETMAX] の範囲内の数値

NL\_SETMAX 定数は、ファイル /usr/include/limits.h で定義されています。数値のセット識別子は、ソース・ファイル内で昇順に現れなければなりません。ただし、番号は連続した値である必要はありません。また、gencat コマンドにより複数のメッセージ・ソース・ファイルを処理して、1 つのメッセージ・カタログを作成するときは、セット識別子の番号は、すべてのソース・ファイルを通して昇順でなければなりません。

- MSGErrors などの、ユーザ定義のシンボリック識別子

シンボリック・セット識別子を指定するときは、mkcatdefs コマンドを使用して、シンボルを gencat コマンドが必要とする数値セット識別子に変換しなければなりません。

セット識別子の後にある文字は、コメントと見なされます。

メッセージ・テキスト・ソース・ファイル内に set ディレクティブがない場合、すべてのメッセージは、省略時のメッセージ・セットに割り当てられます。省略時のメッセージ・セットの番号は、ファイル /usr/include/nl\_types.h 内の NL\_SETD 定数で定義されています。プログラムから catgets() 関数を呼び出して、set ディレクティブが含まれていないソースから作成されたカタログからメッセージを取り出すときは、NL\_SETD 定数をセット識別子として指定します。

---

#### 注意

---

メッセージ・テキスト・ソース・ファイルの set ディレクティブで NL\_SETD を指定したり、1 つのメッセージ・カタログ内に省略時のメッセージ・セットとユーザ定義のメッセージ・セットを混在させないようにします。そのような操作をすると、mkcatdefs や gencat ユーティリティでエラーが発生する可能性があります。また、NL\_SETD 定数に割り当てられる値はベンダに

よって定義されているため、メッセージ・テキスト・ソース・ファイル内で `NL_SETD` をシンボリック識別子として使用すると、`mkcatdefs` 出力の移植性が損なわれます。

---

次に、既存のメッセージ・カタログからメッセージ・セットを削除するエントリについて説明します。カタログの保守に関する一般的な説明については、3.4.3 項を参照してください。

メッセージ・テキスト・ソース・ファイルには、既存のメッセージ・カタログからメッセージ・セットを削除するための `delset` ディレクティブを含めることができます。 `delset` ディレクティブの形式は次のとおりです。

```
$delset n [ comment]
```

変数 `n` は、`gencat` コマンドに指定する、既存のカタログ内のセット番号でなければなりません。 `set` ディレクティブとは異なり、`delset` ディレクティブではシンボリック・セット識別子は指定できません。メッセージ・ファイルを `mkcatdefs` コマンドで前処理するときは、シンボリック識別子を、`mkcatdefs` ユーティリティが割り当てたセット番号とメッセージ番号に対応付ける、別のヘッダ・ファイルをオプションとして生成できます。後でメッセージ・セットを削除したいときは、まずこのヘッダ・ファイルを参照して、削除したいセットのシンボリック識別子に対応する番号を見つけます。次にその番号を `delset` ディレクティブに指定して、セットを削除します。

メッセージ・テキスト・ソース・ファイル `appl.msg` を使用するアプリケーションから、プログラム・モジュール `a_mod.c` を削除するとします。また、`a_mod.c` だけが使用するメッセージは、シンボリック識別子が `A_MOD_MSGS` であるメッセージ・セットに含まれており、`appl_msg.h` ファイルは次の定義文を含むものとしてします。

```
:
:
#define A_MOD_MSGS 2
:
```

この場合、対応する `delset` ディレクティブは、次のようになります。

```
$delset 2    Removing A_MOD_MSG set for a_mod.c in appl.cat.
```

`delset` ディレクティブは、`delset` 単独でソース・ファイル中に指定することも、`delset` ディレクティブと `set` ディレクティブの両方を含む、

より一般的なメッセージ・ソース・ファイルの一部として指定することもできます。後者の場合には、複数のディレクティブが指定子に従って昇順に現れるようにします。

上記の例のディレクティブが単独でソース・ファイル `kill_mod_a_msgs.msg` に含まれ、既存のメッセージ・カタログはディレクトリ `/usr/lib/nls/msg` にあるものとします。この場合、次の `ksh` ループを使用すると、すべてのロケールのカタログ内の、該当するメッセージ・セットを削除できます。

```
for i in /usr/lib/nls/msg/*/appl.cat
do
    gencat $i kill_mod_a_msgs.msg
done
```

### 3.1.3 メッセージ・エントリ

メッセージ・エントリの形式は次のとおりです。

`msg_id message_text`

`msg_id` は、次のいずれかです。

- `[1 - NL_MSGMAX]` の範囲内の数値  
`NL_MSGMAX` 定数は、ファイル `/usr/include/limits.h` に定義されています。メッセージ番号は、直前の `set` ディレクティブで定義されているメッセージ・セットに関連付けられます。`set` ディレクティブが前にない場合は、ファイル `/usr/include/nl_types.h` に定義されている `NL_SETD` 定数が示す省略時のメッセージ・セットに関連付けられます。  
メッセージ番号は、メッセージ・セット内で昇順に並んでいなければなりません。ただし、番号は連続した値である必要はありません。メッセージ番号がセット内で昇順になっていない場合、そのソース・ファイルからメッセージ・カタログを作成しようとすると、`gencat` コマンドはエラーを返します。
- `ERR_INVALID_ID` などの、ユーザ定義のシンボリック名  
メッセージ・テキスト・ソース・ファイルがシンボリック名を含んでいるときは、`mkcatdefs` コマンドを使用して、シンボル名を `gencat` コマンドで処理できる数値に変換しなければなりません。

`message_text` は、プログラムが `msg_id` を使用して参照する文字列です。メッセージ・エントリが現れる前に、`quote` ディレクティブによってクオート文字が使用可能になっている場合、その文字列はクオート文字で



囲むことができます。メッセージ・テキストをクォート文字で囲む利点については、3.1.1 項を参照してください。quote ディレクティブの規則については、3.1.4 項で説明します。

`message_text` の合計長は、`NL_TEXTMAX` 定数で定義されている最大バイト数を超えることはできません。NL\_TEXTMAX 定数は、ファイル `/usr/include/limits.h` に定義されています。

次に、既存のメッセージ・カタログから特定のメッセージを削除するエントリについて説明します。メッセージ・カタログの保守に関する一般的な説明については、3.4.3 項を参照してください。

既存のメッセージ・カタログから特定のメッセージを削除するには、削除したいメッセージの識別子だけからなる行を入力します。このタイプのエントリを使用すると、以降のメッセージの順序に影響を与えずに、メッセージを削除できます。メッセージの削除が正しく実行されるように、次のガイドラインに従ってください。

1. 数値のメッセージ識別子を指定します。

メッセージ・テキスト・ソース・ファイル内でシンボリック識別子を使用している場合は、`mkcatdefs` コマンドで最後にソース・ファイルを処理したときに生成されたメッセージ・ヘッダ・ファイルから、識別子に対応する番号を取得できます。`delset` ディレクティブを使用してメッセージ・セットを削除する場合とは異なり、メッセージの削除にシンボリック・メッセージ識別子を使用しても、`mkcatdefs` はエラーを生成しません。ただし、このシンボルの前に、カタログ内と同じ数のメッセージ・エントリがない場合は、誤ったメッセージを削除してしまいます。

2. 識別子の後には改行以外の文字を置くことはできません。`msg_id` の後にスペースやタブの区切り文字が続く場合、メッセージは削除されず、メッセージ・テキストが空文字列に変更されます。

3. カタログにユーザ定義のメッセージ・セットが含まれている場合は、メッセージを削除するエントリの前に、適切な `set` ディレクティブがあることを確認します。適切な `set` ディレクティブがない場合は、誤ったメッセージ・セットからメッセージが削除されることがあります。ステップ 1 で説明したメッセージ識別子と同じ理由で、`set` ディレクティブにはシンボリック・セット識別子ではなく、数値セット識別子を使用します。

4. セット内のすべてのメッセージを置き換えるのでなければ、`gencat` だけを使用してファイルを処理します。セット内のすべてのメッセージを置き換えるには、`mkcatdefs` ユーティリティを使用します。このユーティリティは、入力ファイル内で指定した個々の `set` ディレクティブの前に `delset` ディレクティブを生成します。これはメッセージ・セット内のすべてのメッセージを置き換えたい場合には有用ですが、削除したいメッセージを 1 つか 2 つだけ入力ファイルに指定した場合には、意図したとおりの結果は得られません。

次の 2 つの例を考えてみます。

- この例は、`gencat` コマンドで処理したメッセージ・テキスト・ソース入力を使用します。この例のコマンドは、メッセージ・セット 2 からメッセージ 5 を削除します。

```
$set 2
5
```

- この例は、同じソース入力を使用します。ただし、この場合、ソースは `mkcatdefs` コマンドで前処理されます。`delset` ディレクティブを追加すると、メッセージ・カタログのセット 2 のすべてのメッセージが削除されます。

```
$delset 2
$set 2
5
```

### 3.1.4 `quote` ディレクティブ

`quote` ディレクティブは、メッセージ・テキスト文字列を囲むためのクォート文字を有効にするか、無効にします。`quote` ディレクティブの形式は次のとおりです。

```
$quote[ character]
```

変数 *character* は、メッセージ文字列の区切り記号として認識される文字です。次の例の `quote` ディレクティブは、メッセージ文字列の区切り記号として二重引用符を指定します。

```
$quote "
```

`quote` ディレクティブを指定しないか、*character* を省略した場合は、メッセージ・テキスト文字列をクォート文字で囲むことはできません。

メッセージ・テキスト・ソース・ファイルには、複数の quote ディレクティブを含めることができます。この場合、各ディレクティブは、ファイル内のそのディレクティブの後にあるメッセージ・エントリに影響します。ただし、メッセージ・ファイルは通常、1 番目のメッセージ・エントリの前に quote ディレクティブを 1 つだけしか含みません。

### 3.1.5 コメント行

\$ で始まり、その後にスペースまたはタブ文字が続く行は、コメントと見なされます。mkcatdefs コマンドと gencat コマンドは、コメント行を無視します。

メッセージ・ファイルは、通常プログラマ以外の人翻訳します。そのため、メッセージを翻訳する人のために、コメント行にはリテラルや置換書式指定子を含む文字列を持つメッセージ・エントリの扱い方について、必ず説明を入れるようにします。次に例を示します。

```
$ Note to translators: Translate only the text that is within
$ quotation marks ("text text text") on a given line.
$ If you need to continue your translation onto the next line,
$ type a backslash (\) before pressing the newline
$ (Return or Enter) key to finish the message.
$ For an example of line continuation, see the
$ line that starts with the message identifier E_COM_EXISTBADGE.
:
:

$ Note to translator: When users see the following message, a badge
$ number appears in place of the %ld directive.
$ You can move the %ld directive to another position
$ in the translated message, but do not delete %ld or replace %ld with
$ a word.
$
E_COM_EXISTBADGE          "Employee entry for badge number %ld \
already exists"
:
:

$
$ Note to translator: The item %2$d/%1$d/%3$d indicates month/day/year
$ as expressed in decimal numbers; for example, 3/28/81.
$ To improve the appropriateness of this date input format, you can change
$ only the order of the date elements and the delimiter (/).
$ For example, you can change the string to %1$d/%2$d/%3$d or
$ %1$d.%2$d.%3$d to indicate day/month/year or day.month.year
$ (28/3/81 or 28.3.81).
$
I_SCR_IN_DATE_FMT          "%2$d/%1$d/%3$d"
:
:
```

3.3 節で説明するように、Tru64 UNIX には翻訳者がメッセージ・ソース・ファイル内の翻訳可能なテキストを素早く見つけ出し、編集できるようにす

るための、trans ユーティリティが用意されています。しかし、このユーティリティが使用できるとしても、メッセージの内容やプログラムの構文について、プログラマが情報を提供しなくてもいいことにはなりません。

### 3.1.6 メッセージ・スタイルのガイドライン

メッセージなどのテキスト文字列を英語で作成するときは、次の点を考慮する必要があります。

- 一般に、英語のテキスト文字列は、他の言語の同等なテキスト文字列よりも短い。英語のテキスト文字列を翻訳すると、その長さは平均で 30 ~ 40 % 増加します。20 文字より短い文字列では、増加する割合はさらに上がります。

以下のガイドラインは、文字列を英語から他の言語に翻訳するときに、文字列が長くなることを考慮しておくためのものです。

- テキスト文字列を 1 行 (たとえば 80 文字) に収めなければならないときは、英語のテキストが、利用可能な領域の半分以下に収まるようにする。可能であれば、長さを制限せずに、テキストを次の行に折り返せるようにします。
- メニューや、書式、スクリーン、ウィンドウを設計するときに、英語のテキストが利用可能なスペースの大半を占めることがないようにする。
- ダイアログ・ボックスは、構成要素を移動できるような形で設計する。アプリケーションを地域化する開発者は、テキスト長が変化するために、ダイアログ・ボックスの内容を再構成しなければならないことがあります。また、アジア系言語では、アジア系文字の入力をサポートしなければならないこともあります。
- グラフィック内にテキストを埋め込まない。テキストがグラフィックに埋め込まれていると、アプリケーションを地域化する際に、グラフィック全体を作り直さなければなりません。また、テキストの翻訳により、グラフィックのサイズが大きくなったり、見栄えが悪くなることがあります。
- 英語以外の言語では名詞が性を持つことがあり、そのため、名詞そのものや、関連する形容詞と動詞のスペルが変化する場合がある。名詞のスペルは、名詞が動詞の主語なのか目的語なのか、あるいは前置詞の目的語なのかなど、使われ方によって変化することがあります。さら

に、動詞の肯定形、否定形、および命令形を作る方法など、英語とは異なる文法規則が存在することがあります。このような条件により、次の規則が存在します。

- 種類の異なる文字列 (たとえば、異なる名詞、形容詞、動詞、あるいは、その組み合わせを表す文字列) を実行時に連結してメッセージを作成しない。

形容詞や動詞が、異なった性を持つ複数の名詞に係る場合、翻訳者は、表示される可能性のあるすべての文に対し、文法的に正しい訳文を作成できないことがあります。その場合、アプリケーションの地域化対応を計る開発者は、エラー処理ロジックを再設計して、アプリケーションが 1 種類だけでなく、複数の異なるメッセージを表示できるようにしなければならないことがあります。

- 同じテキスト変数を複数の文字列に挿入するときは、十分な注意が必要になる。各文字列の意味が文法上異なる場合、単語のスペルを変えなければならないことがあります。また、英単語と、他の言語の単語が 1 対 1 に対応しているとはかぎりません。たとえば英語では、“not” を含むテキスト変数を動詞句に含めることにより、否定文が作れます。しかしこのメッセージは、フランス語には翻訳できません。フランス語では通常、意味を否定するためには、動詞の前に “ne”、動詞の後に “pas” の 2 つの単語が必要になります。

一般に、パス名、ファイル名、および完全な文の文字列は、他の文字列に挿入しても問題ありません。

- “None” という単語を、ボタン・ラベルやメニュー項目として使用しない。None が係る語句に複数の性がある場合、この単語は翻訳できないことがあります。
- 一般的に、完全な文としてメッセージを作成する。言語によって文法規則が異なるため、断片的なメッセージを作成すると、翻訳上の問題が発生することがあります。

メッセージが、システム・エンティティを示す構成要素 (コマンド、ユーティリティ、エラー重大度、サーバなど) と、情報テキストまたはエラーテキストを含む別の構成要素からなっている場合、動詞で始まるメッセージに関する規則を破ることができます。この場合、メッセージ構成要素の構築方法や、メッセージで参照しているシステム・エンティティについての、翻訳者へのコメントを、

メッセージ・ソース・ファイルに含めるようにしてください。また、情報テキストやエラー・テキスト構成要素には、文法的に完全なフレーズを使用してください。メッセージ・ソース・ファイルへの構成要素の追加については、3.1.5 項を参照してください。

- 動詞でメッセージを開始しない(“you” が主語であることが明らかな命令形のメッセージの場合を除く)。

次のメッセージは、翻訳者が文の主語を判断できないか、ローカル言語での動詞の正しい形式を判断できないため、一部の言語には翻訳できません。

Is a directory.

Could not open file.

- 単語の先頭の数文字に基づく識別子は、翻訳後に一意であるとはかぎらない。たとえば、複数の項目から 1 項目を選択するようにユーザに求めるときは、1 文字を項目識別子として受け付けるのがアプリケーションにおける一般的な手法です。このときに、選択する文字が、項目名の最初の 1 文字や数文字でなければならないという要件は不適切です。項目識別子は、翻訳者が任意の文字や番号に置き換えられるものでなければなりません。
- 言語によっては、翻訳者が語順を変更しなければならない文法を持つことがある。翻訳者が、ローカル言語の文法規則に合わせてメッセージの構成要素の語順を変更できるように、2.4.2 項で説明している置換指定子を使用します。
- 不明確な、あいまいな、あるいは電報で使われるような言葉づかいを含むメッセージは、誤訳される可能性がある。正しく翻訳されるように、次のガイドラインに従ってください。
  - メッセージ・ファイルに、プログラム・ソース・ファイルの場合と同様のドキュメントを含める。文の構造を説明したり、その言語を母国語としない人が間違いやすい言い回しを明確化するコメントを用意してください。
  - 冠詞 (the, a, an) や動詞の “to be” 形を適切に使用する。プログラマは、メッセージ文字列のサイズを小さくするために、これらの単語を省略することがあります。しかし、これらの単語を省略してしまうと、名詞と動詞、主部と述部、能動態と受動態の区別が

付きにくくなります。“Maximum parameter count exceeded” というメッセージは、この問題の一例です。

- “can’t” や “don’t” などの一般的な短縮形は使用してもかまわないが、“should’ve” などの一般的でない短縮形は使用しない。行の長さを短くするために英語で短縮形を使用しても、翻訳結果が同じように短縮できるとはかぎりません。
- プログラムが変数名やプログラムのコメントでよく使用する省略形の使用は避ける。特に、pkt, msg, tbl, ack, max などの使用は避けてください。これらの略語は辞書には載っておらず、翻訳者がその意味を推測しなくてはならないことがあります。ただし、製品名およびユーティリティ名の正式な略語や、標準の名前、プロトコルの名前など、市販の文献に載っている略語 (ANSI や TCP/IP など) は使用してもかまいません。
- 文法的に正しい単語を使用する。英語を母国語とする人は、既存の名詞から新しい動詞や形容詞を、また既存の動詞から新しい名詞を作り出す傾向があります。これらは、特に意図した用法が英語の辞書に載っていない場合には、翻訳者を混乱させます。たとえば、“Invalid parameter delimiter.” というメッセージでは、“parameter” という単語が形容詞として使用されています。
- 俗語や、意図する意味が辞書に載っていない単語の使用は避ける。俗語は、他の言語では対応する単語がないか、誤解される可能性があります。たとえば、“Server hang” というメッセージは、ソフトウェアの開発やシステムの管理を行っている、英語を母国語とする人には理解できるものですが、他の言語ではその意味は、「システムがウェ이터をリンチした」に変わってしまうことがあります。“The %s server failed.” というメッセージの方が正しく翻訳されます。
- 通常は、メッセージ・ファイル内では位置フォーマット要素を使用してください。ただし、メッセージ内にフォーマット・フラグが 1 つしかない場合、位置要素の値を指定しないため、翻訳者が混乱することがあります。位置フォーマット要素を含むメッセージ・ファイルには、意図した結果を翻訳者が理解できるように、コメントを十分に含める必要があります。

## 3.2 既存のプログラムからのメッセージ・テキストの抽出

既存のプログラムを国際化するときには、メッセージ文字列を抽出してメッセージ・ソース・ファイルに格納し、メッセージ・カタログからメッセージを取り出すように呼び出しを変更する、Tru64 UNIX のツールが利用できます。

ツール	説明
extract コマンド	プログラム・ソース・ファイルからテキスト文字列を対話形式で抽出し、抽出した文字列をソース・メッセージ・ファイルへ書き込みます。またこのコマンドは、抽出された各文字列を、 <code>catgets()</code> 関数の呼び出しに置き換えます。
strextact コマンド	文字列をバッチ処理で抽出します。
strmerge コマンド	<code>strextact</code> で作成されたメッセージ・ファイルから文字列を読み取り、プログラム・ソース内でこれらの文字列を <code>catgets()</code> 関数の呼び出しに置き換えます。

次の呼び出しを考えてみます。

```
printf("Hello, world\n");
```

`extract` コマンドを使用するか、あるいは、`strextact` コマンドと `strmerge` コマンドを続けて使用すると、以下の処理を実行できます。

- メッセージ・テキスト・ソース・ファイルへ以下のエントリを書き出す ("Hello, world" が最初に抽出された文字列とする)。

```
$set 1
$quote "
1 "Hello, world\n"
```

- `printf()` の呼び出しを次のように変更する。

```
printf(catgets(cat, 1, 1, "Hello, world\n"));
```

コマンドへの入力が `prog.c` というプログラム・ソース・ファイルだとすると、コマンドは、`prog.msg` (メッセージ・テキスト・ソース・ファイル)、`nl_prog.c` (プログラム・ソースの国際化バージョン)、および `prog.str` (他のユーティリティから参照できる中間文字列ファイル) という 3 つのファイルを作成します。このコマンドは、入力ソース・プログラムに加え、次のファイルを使用します。

- パターン・ファイル



このファイルは、抽出コマンドがプログラム中の文字列を見つけるために使用する、パターンを指定します。ユーザ独自のパターン・ファイルを指定することもできます。省略時には、抽出コマンドはファイル `/usr/lib/nls/patterns` を使用します。

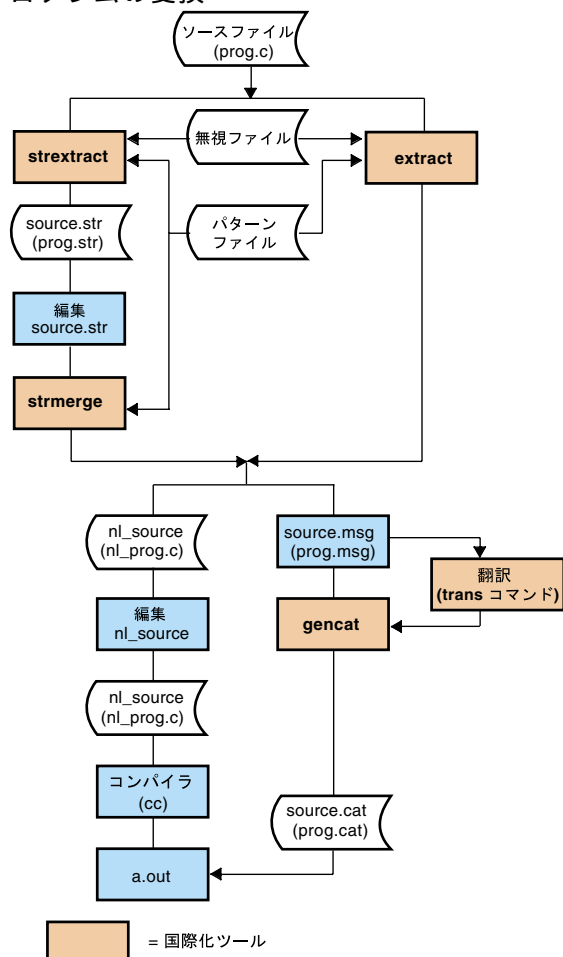
- オプションの無視ファイル (省略可能)

このファイルには、抽出コマンドが無視する文字列を指定します。

`extract`、`strextact`、および `strmerge` コマンドは、プログラムの国際化に必要なすべての変更を行うわけではありません。たとえば、変更されたプログラム・ソースを手作業で編集し、`setlocale()`、`catopen()`、および `catclose()` の呼び出しを追加する必要があります。さらに、マルチバイト文字変換ルーチンを追加したり (アジア系ロケールの場合)、ユーザ定義ルーチンを改善して、ルーチンがメッセージ・カタログや `langinfo` データベースで定義されている値に応じてその動作を変更するようにしなければならないことがあります。

図 3-1 に、既存のプログラムを変更して、メッセージ・カタログを使用できるようにするためのファイルとツールを示します。`extract` コマンド、`strextact` コマンド、および `strmerge` コマンドの使用方法については、`extract(1)`、`strextact(1)`、`strmerge(1)`、および `patterns(4)` のリファレンス・ページを参照してください。

図 3-1: メッセージ・カタログを使用できるようにするための、既存のプログラムの変換



ZK-0045U-AIJ

### 3.3 メッセージ・ソース・ファイルの編集と翻訳

メッセージ・テキスト・ソース・ファイルの編集には、次の要件を満たす任意のテキスト・エディタが使用できます。

- 必要な文字が入力デバイスで生成できること。
- 8ビット文字やマルチバイト文字を使用するときに、エディタがこのデータを透過的に処理できること。

入力デバイスの要件は、西ヨーロッパ言語以外の言語では、地域化されたソフトウェア・サブセットに付属するターミナル・ドライバ、ロケール、フォントなどのコンポーネントによって満たされます。

`ed`、`ex`、および `vi` エディタは、8 ビット・データやマルチバイト・データを透過的に処理するという要件を満たしています。地域化されたソフトウェア・サブセットには、8 ビット文字やマルチバイト文字を処理できる拡張版のエディタ、たとえば Emacs などが含まれていることがあります。

Tru64 UNIX オペレーティング・システムは、メッセージ・テキスト・ソース・ファイルを別のロケールに翻訳する人のために、`trans` コマンドを提供しています。このコマンドにはマルチウィンドウ機能があり、ファイルのオリジナル・バージョンと翻訳バージョンの両方が表示できます。さらにこのコマンドは、ファイル内の翻訳可能な文字列を自動的にたどる機能も備えています。詳細については、`trans(1)` のリファレンス・ページを参照してください。

メッセージの翻訳が正しく行われるように、メッセージ・テキスト・ソース・ファイルに含めるコメントの例については、3.1.5 項を参照してください。

翻訳されたメッセージ・テキスト・ソース・ファイルの例を参照するには、ディレクトリ `/usr/examples/i18n/xpg4demo/` で `*.msg` ファイルを検索してください。

```
% cd /usr/examples/i18n/xpg4demo/
% ls *.msg
:
```

翻訳されたメッセージ・カタログは特定のロケールおよびエンコーディング・フォーマットに対応付けられています。多くの言語で、複数のロケールおよびエンコーディング・フォーマットをサポートしています。このため、このような言語のメッセージ・カタログは、複数のエンコーディング・フォーマットで用意する必要があります。メッセージ・ソース・ファイルのコードセット変換にはコンバータが利用できますが、1 つの言語に対して複数のメッセージカタログを作成しインストールするのは効率的とはいえません。このため、`catopen()` および `catgets()` 関数は、メッセージ・カタログの動的なコードセット変換をサポートしています。 `/usr/share` ディレクトリの `.msg_conv-locale_name` ファイルのセットが、メッセージ・カ

タログのコードセット変換を制御します。詳細については、catopen(3) を参照してください。

### 3.4 メッセージ・カタログの生成

gencat コマンドは、1 つまたは複数のメッセージ・テキスト・ソース・ファイルからメッセージ・カタログを生成します。メッセージ・セット、メッセージ・エントリ、あるいはその両方において、ソース・ファイルに数値識別子ではなくシンボル識別子が含まれている場合は、mkcatdefs コマンドを使用して、それらのソース・ファイルを前処理しなければなりません。例 3-2 に、シンボリック識別子を持つメッセージ・テキスト・ソース・ファイルを、省略時のロケールと非省略時のロケールについて、対話形式で処理する例を示します。この例の内容に基づいて、以降の項で各コマンドについて説明します。

#### 例 3-2: 対話型操作によるメッセージ・カタログの生成

```
% mkcatdefs xpg4demo xpg4demo.msg | gencat xpg4demo.cat 1
mkcatdefs: xpg4demo_msg.h created 2
% setenv LANG fr_FR.ISO8859-1 3
% mkdir fr_FR 4
% mkcatdefs xpg4demo xpg4demo_fr_FR.msg -h | gencat \
fr_FR/xpg4demo.cat 5
mkcatdefs: no msg.h created 6
```

1 mkcatdefs コマンドには次の名前を指定します。

- ヘッダ・ファイル用のルート名

このヘッダ・ファイルは、プログラムで使用するシンボリック識別子を、メッセージ・カタログ内の数値に対応させます。

- 処理するメッセージ・テキスト・ソース・ファイルの名前

前処理されたメッセージ・ソースは、パイプを通して gencat コマンドに渡されます。gencat コマンドには、メッセージ・カタログの名前を指定します。

2 mkcatdefs コマンドは、作成したヘッダ・ファイルの名前を標準エラー出力に出力します。このユーティリティは、ルート名に \_msg.h を付加して、ヘッダ・ファイルの名前とします。

- ③ 非省略時のロケール用のメッセージ・ファイルを生成するときは、`LANG` 環境変数に、メッセージ・カタログがサポートするロケール名を設定しなければなりません。ここでは、`fr_FR.ISO8859-1` です。
- ④ プログラムからオープンされるメッセージ・カタログの名前は、ロケール名が違って同じです。そのため、メッセージ・カタログを格納するディレクトリをロケールごとに作成しなければなりません。
- ⑤ この行は、地域化されたメッセージ・カタログを作成します。  
`mkcatdefs` ユーティリティによって作成されるヘッダ・ファイルは、どのロケールでも同じです。省略時のメッセージ・カタログ用のヘッダ・ファイルは、すでに作成されています。そのため、この `mkcatdefs` コマンドでは `-h` フラグを指定して、新たなヘッダ・ファイルを作成しないようにします。`gencat` コマンドに指定されたカタログは、一時的なロケール・ディレクトリに置かれます。ユーザ・システムでは、このバージョンのカタログは、省略時ディレクトリの `/usr/lib/nls/msg/fr_FR.ISO8859-1` に移動するか、アプリケーション固有のディレクトリに移動できます。
- ⑥ `mkcatdefs` コマンドは、ヘッダ・ファイルが意図したとおりに作成されなかったことを通知します。

アプリケーションを構築するための `makefile` に、メッセージ・カタログの生成を統合する例については、`/usr/examples/i18n/xpg4demo/Makefile` ファイルを参照してください。

### 3.4.1 `mkcatdefs` コマンドの使用法

`mkcatdefs` コマンドは、1 つまたは複数のメッセージ・ソース・ファイルを前処理して、シンボリック識別子を数値定数に変換します。このユーティリティは、次の機能を備えています。

- 前処理されたメッセージ・ソースを標準出力に出力する。このため、例 3-2 で説明するように、パイプを通して出力を `gencat` コマンドへ渡すことも、リダイレクション指示子 (`>`) を使用して出力をファイルへ書き出すこともできます。
- 新しいメッセージ・カタログ内のメッセージ・セットとメッセージを識別する番号を、ソース・プログラムで参照されているシンボリック識別子に対応付けるヘッダ・ファイルを作成する。

このヘッダ・ファイルは、このカタログをオープンし、シンボリック識別子を使用するメッセージ・セットとメッセージを参照する、すべてのプログラム・モジュールにインクルードしなければなりません。

シンボリック識別子の利点は、メッセージ・セット識別子やメッセージ識別子の引数を含む呼び出しをコーディングする際に、番号の代わりにシンボリック識別子を指定できることです。シンボリック識別子を使用することにより、プログラム・ソース・コードが読みやすくなり、また、メッセージ・カタログ内でのセットとメッセージ・エントリの順序にコードが依存しなくなります。mkcatdefs ユーティリティは、メッセージ・テキスト・ソース・ファイル进行处理するたびに、セットとメッセージ・シンボルを番号に対応付けるヘッダ・ファイルを作成します。そのため、メッセージ・ファイルの変更が完了すれば、プログラムは、新しいヘッダ・ファイルを使用してコンパイルするだけで更新できます。

---

#### 注意

---

mkcatdefs コマンドには、この章でまだ説明していないオプションが2つあります。

-s オプションを使用すると、gencat コマンドへ渡される出力で、シンボリック名のサポートが有効になります。dspmsg コマンド(シェル・スクリプト内で使用)にはこれに対応する -s オプションがあり、このサポートを使って構築されたメッセージ・カタログから、シンボリック名を用いてメッセージを取得できるようになっています。(libc ライブラリの catgets() 関数は、X/Open UNIX 標準の XSH 仕様によって、実行時に、シンボルではなく数値の識別子を用いてカタログからメッセージを取得するように制限されています。)

-m オプションを指定すると、デフォルト・メッセージ文字列が自動的に生成され、シンボリック名が割り当てられます。この機能により、dspmsg コマンドや catgets() の呼び出しに、カタログからメッセージが取得できない場合に表示するデフォルト・メッセージ文字列を指定する必要がなくなります。

これらのオプションについての詳細は、mkcatdefs(1) を参照してください。

---

メッセージ・セットとカタログ用のシンボリック識別子を定義する機能は、XSH 仕様には含まれていません。このため、この仕様に準拠するすべてのオペレーティング・システムで `mkcatdefs` コマンドが利用できるわけではありません。ただし、`mkcatdefs` コマンドを使用して作成されたソース・テキスト・メッセージ・ファイルとヘッダ・ファイルは、この仕様に準拠するシステム間でポータビリティがあります。

`mkcatdefs` コマンドは、現在処理しているメッセージ・ソース入力ストリーム中のシンボルの位置に基づいて、シンボル識別子に番号を割り当てます。既存のカタログを変更するときは、`mkcatdefs` コマンドへのソース入力で指定するシンボルと、それらのシンボルに対応する、既存のメッセージ・カタログ内の数値を正しく対応させてください。

一般に、`mkcatdefs` ユーティリティは、メッセージ・カタログの一部を変更するときに使用するツールではなく、メッセージ・カタログ全体を再度作成するためのツールです。次のガイドラインに従ってください。

- メッセージとメッセージ・セットを削除するときは、メッセージ・ソース・ファイル内の適切な位置に、シンボル識別子ではなく数値識別子を指定する。これにより、メッセージ・セットや個々のメッセージの削除が以降のエントリの順序に影響しなくなる。
- 新しいセットは、入力ソース・ストリームの終わりに定義する。カタログが複数のソース・ファイルで生成されている場合、入力ソース・ストリームの終わりは、最後のソース・ファイルの終わりになります。
- 既存のメッセージ・セットに新しいメッセージを追加するときは、そのセットの終わりに定義する。
- カatalog全体のソース・エントリを指定する。Catalog全体のエントリを指定しないと、`mkcatdefs` は完全なメッセージ・ヘッダ・ファイルを作成しません。既存のシンボルと新しいシンボルの両方を使用してメッセージを識別するプログラムをコンパイルするには、完全なヘッダ・ファイルが必要です。また `mkcatdefs` は、入力ソースで指定した各 `set` ディレクティブの前に、`delset` ディレクティブを生成します。つまり `mkcatdefs` は、指定されたセット内のすべてのメッセージが入力によって完全に置き換えられることを前提としています。
- カatalogが複数のソース・ファイルから生成されている場合は、既存のCatalogを作成したときに指定したのと同じ順序でソース・ファイルを指定する。違う順序を指定すると、そのCatalogをオープンする

すべてのプログラム・モジュールのコンパイル時に使用したヘッダが無効になります。プログラムのコンパイル時に使用したメッセージ・ヘッダ・ファイル内のシンボルと番号の対応付けを無効にしないかぎり、新しいメッセージを参照しないプログラムは、再コンパイルする必要はありません。

- メッセージ・テキスト・ソース・ファイルの `set` ディレクティブに `NL_SETD` を指定したり、同じメッセージ・カタログ内に省略時のメッセージ・セットとユーザ定義のメッセージ・セットを混在させない。このようにすると、`mkcatdefs` ユーティリティや `gencat` ユーティリティでエラーが発生することがあります。
- `mkcatdefs` ユーティリティが、メッセージ識別子とメッセージ・テキストの間の複数のスペースを、スペース 1 文字に圧縮することに気を付ける。この変換により、UNIX 標準との互換性が確保され、他の UNIX や LINUX プラットフォームの要件が満たされます。

### 3.4.2 gencat コマンドの使用法

`gencat` コマンドは、1 つまたは複数のメッセージ・テキスト・ソース・ファイルをマージして、メッセージ・カタログを生成します。次に例を示します。

```
# gencat en_US/test_program.cat test_program_en_US.msg
```

`gencat` コマンドは、カタログ・パスで指定されたカタログが存在しない場合、メッセージ・カタログを新規に作成します。カタログが存在する場合、このコマンドは指定されたメッセージ・テキスト・ソース・ファイル (複数ファイルも可) を使用して、そのカタログを変更します。`gencat` コマンドは、メッセージ・ソース・データを標準入力から読み込むこともできます。このため、`mkcatdefs` コマンドなどの別のプログラムの出力をパイプを通して `gencat` へ渡すときは、ソース・ファイルの引数を省略できます。

X/Open の UNIX 標準では、メッセージ・ソース・ファイルやカタログ・ファイルのファイル名拡張子は規定されていません。通常、Tru64 UNIX システムでは、ソース・ファイルには `.msg`、カタログには `.cat` を拡張子として使用します。`gencat` コマンドで作成されたメッセージ・カタログはバイナリ・データとしてエンコードされているため、エンコード・タイプの異なるシステム間ではポータビリティがありません。`mkcatdefs` コマンドで前処理されたメッセージ・テキスト・ソース・ファイルは、X/Open UNIX CAE Specifications に準拠するシステム間でポータビリティがあります。



詳細については、`gencat(1)` のリファレンス・ページを参照してください。

### 3.4.3 メッセージ・カタログの設計と保守に関する留意点

メッセージ・セットとメッセージ・エントリは、メッセージ・カタログの 1 つのバージョン内での順序を表す番号により、実行時に識別されます。既存のカタログ内のメッセージ・セットやエントリを追加したり削除する場合、メッセージを識別する順序指定子を変えないように注意が必要です。

"Enter street address:" という英文テキストのメッセージが、メッセージ・カタログのオリジナル・バージョンの生成時には 3: 10 (3 番目のメッセージ・セットの 10 番目のメッセージ) で識別されたとします。 `gencat` コマンドに入力されるメッセージ・ソース・ファイルで次のいずれかの操作を行った場合、新しいバージョンのカタログではこのメッセージの識別番号が変わります。

- 入力ソースの先頭にメッセージ・セットを挿入する。
- 3 番目のメッセージ・セットで、エントリ "Enter street address:" の前にメッセージを挿入する。
- 3 番目のメッセージ・セットで、エントリ "Enter street address:" より前のメッセージを、メッセージ削除ディレクティブ (メッセージ番号だけを指定したディレクティブ) を指定せずに削除する。

次の 2 つのプログラム・セグメントのように、順序についての制限事項を翻訳者に対して説明するコメントを、コードに追加することを考慮してください。

```
$ Note - Do not reorder message descriptors for columns.
```

```
S_COM_LIST_ROW          "%5d          %20s      %20s      %4s      %9s\n"
```

```
$ The first descriptor must always be displayed at the beginning of error messages.
```

```
$ The second descriptor contains the first name.
```

```
$ The third descriptor contains the surname.
```

```
S_COM_LIST_ERROR        "%1$s: Error badge number for %2$s      %3$s incorrect\n"
```

メッセージを数値識別子で参照するプログラム・ソースの場合、メッセージ・セットやメッセージ・エントリの位置を変更すると、メッセージを参照するプログラム呼び出しも変更しなければなりません。メッセージをシンボリック識別子で参照するプログラム・ソース・ファイルの場合、エントリ位置の変更に伴う保守作業は、各モジュールで大幅に低減されます。つまり、どのプログラム・モジュールも、`mkcatdefs` ユーティリティによって生成さ

れた新しいヘッダ・ファイルでコンパイルすることにより、メッセージ・カタログの新しいバージョンに同期させることができます。

プログラム・ソースをコンパイルして、新しいメッセージ・カタログに同期させる機能を使用しても、同じメッセージ・カタログを複数のソース・ファイルが参照するような複雑なアプリケーションの問題には対処できません。そのようなアプリケーションでは、モジュールごとの保守作業を可能にすることが一般的な目標となります。つまり、アプリケーションが一旦エンド・ユーザ・サイトにインストールされた後は、アプリケーションの全モジュールを再コンパイルして再インストールするのではなく、個別のモジュールとそれに関連するメッセージ・カタログの更新が可能でなければなりません。この目標を達成するためには、さまざまな設計手法が考えられますが、以下に説明する設計例を参考にして、サイトでのアプリケーションの開発と保守に最適なメッセージ・システムの設計方針を決定してください。

- プログラム・モジュールごとに、1つのメッセージ・ソース・ファイルとカタログ

- 利点

プログラマが1つのソースを共有する場合に発生する問題がないため、個々のプログラマにとって実現するのが最も簡単な方法です。リビジョン・コントロール・システム (RCS) やソース・コード・コントロール・システム (SCCS) などのソース・コントロール・ソフトウェアは、複数のプログラマによって保守されているファイルを管理する際の有用なツールです。しかし、場合によっては、アプリケーションの複数のバージョンに対してプログラマが平行して作業を進めることもあります。このように複雑さが一段階増すと、管理はますます難しくなります。メッセージ・ソース・ファイルと関連プログラム・ソースを1対1に対応させることができれば、特定の時点で特定のリリース・サイクルのアプリケーションを構築するためには、メッセージ・ファイルのどこを変更すればよいかが簡単に判定できます。

メッセージ・カタログがモジュールに固有であれば、エンド・ユーザ・サイトに新しいバイナリ・モジュールをインストールする際に、メッセージ・カタログ全体を置き換えることができます。モジュールの置き換えにより、同じアプリケーションの他のモジュールの実行時動作に影響を与える可能性が最小限になります。

- 欠点

アプリケーションは実行時に、モジュールの数だけメッセージ・カタログをオープンし、クローズしなければならないことがあります。メッセージ・カタログのオープンには、性能上のオーバーヘッドが伴います。また、メッセージ・カタログのオープンによって、ユーザ・プロセスとシステム用のオープン・ファイル・テーブルに割り当てられるオープン・ファイル記述子の数が増加します。同時にオープンできるファイルの数には、システム全体の最大値とプロセスごとの最大値があり、これらの制限値はシステムによって異なります。

Tru64 UNIX システムでは、メッセージを取り出す性能を向上するために、オープンされたメッセージ・カタログはメモリにマップされます (ファイルはクローズされます)。この機能により、複数のメッセージ・カタログを同時にオープンした場合でも、オープン・ファイル数の制約にはほとんど影響されることはありません。ただし、アプリケーションの移植先となる他のプラットフォームでは、このような機能がサポートされていないこともあります。

- プログラム・ソースごとに 1 つのメッセージ・ソース・ファイル、アプリケーションごとに 1 つのカタログ

- 利点

この方法を使用すると、前述の、プログラムごとに 1 つのメッセージ・ソース・ファイルとカタログを用意する場合と同じ利点があります。さらにこの設計では、カタログを 1 つにすることにより、Tru64 UNIX 以外のシステムにアプリケーションを移植する場合でも、オープン操作に関連するさまざまな問題を解消できます。

- 欠点

複数のソース・ファイルから 1 つのメッセージ・カタログを作成する場合、メッセージ・セット・ディレクティブを慎重に扱わないと、保守上の問題が発生することがあります。1 つのソース・ファイルに割り当てるセット数を固定にすることが、最善の規則となります。たとえば、エラーに 1 セット、情報表示に 1 セット、その他の文字列に 1 セットというように割り当てます。プログラマが、メッセージ・ソース・ファイルのバージョンごとに、メッセージ・セットの数を変更できるようにすると、後続のプログラ

ム・モジュールのメッセージ・セット番号がカタログのバージョンによって変わる可能性があります。つまり、ソース・コードが変更されていない他のモジュールも、メッセージ・カタログの新しいバージョンと同期をとるためだけに、アップデート・リリースに含めなければならないことがあります。

ソース・ファイルにメッセージ・セットが定義されていなかったり、一部のソース・ファイルだけにメッセージ・セットが定義されている場合にも、同じような保守上の問題が発生します。

`mkcatdefs` コマンドと `gencat` コマンドは、入力ソース・ファイルを連結するので、最後の入力ソース・ファイルの終わりだけにファイルの終わりを示すマークが存在することになります。このため、セットがいずれのファイルにも定義されていない場合、すべてのメッセージは省略時のメッセージ・セットの一部であると見なされます。プログラム内の呼び出しでは、`NL_SETD` 定数が省略時のメッセージ・セットを示します。セットが定義されていない場合、最後のソース・ファイル以外のファイルにメッセージを追加すると、入力ストリーム上の後続のすべてのソース・ファイル内のメッセージの数値識別子が変わります。

さらに、複数のソース・ファイルを1つのメッセージ・カタログにまとめる設計では、生成されるメッセージ・カタログが非常に大きく、メモリ容量に制限がある場合には、上記以外の欠点も顕著になります。すでに説明したように、メッセージを取り出す際のディスク I/O によって性能が低下しないように、メッセージ・カタログはオープンされた時点でメモリにマップされます。アプリケーションを実行するユーザが、利用可能なモジュールの一部に対応するソフトウェアとメッセージ以外は使用しない場合、モジュールごとにメッセージ・カタログを用意する設計の方が、メッセージが特定の実行サイクルでオープンされたときに使用されるメモリの合計容量を節約できます。

また、一部のメッセージ・ソース・ファイルだけにメッセージ・セットが定義されている場合は、メッセージ・セットがソース・ファイルの境界を超えてしまうことがあります。入力ストリーム上の後続のソース・ファイルで定義されたメッセージは、先に処理されたソース・ファイルで定義されたメッセージ・セットの一部であると見なされます。またこの配置では、新しいメッセージが別の

ソース・ファイルに追加された場合でも、メッセージ・エントリの位置が変わることがあります。

- 組み合わせる際の方針

アプリケーションによっては、モジュールごとに用意された複数のソース・ファイルから作成される1つ以上のメッセージ・カタログと、すべてのプログラマによって保守される1つのソース・ファイルから作成されたいくつかのメッセージ・カタログを持つ設計の方がよい場合があります。

たとえば、アプリケーション内の多くのモジュールが同じエラー状態に対するメッセージを生成する場合、メッセージ・テキストの一貫性が重要な目標になります。そのような場合は、エラー・メッセージが定義されている1つのメッセージ・テキスト・ソース・ファイルを使用して、メッセージ・カタログを1つ作成します。このソース・ファイルを使用して、エラー用のメッセージ・セット、警告用のメッセージ・セットなどを定義できます。この場合、プログラマに対して、新しいメッセージは各セットの末尾だけに追加し、不要になったメッセージはメッセージ削除ディレクティブを使用して削除するように指示します。メッセージ削除ディレクティブを使用すると、同じセット内の後続メッセージの位置を変更せずに、カタログからメッセージを削除できます。

メッセージ・ファイルの保守作業を簡単にするには、次のガイドラインを考慮してください。

- 新しいメッセージは、メッセージ・セットの最後に追加する。これにより、既存の古いメッセージ・カタログとの互換性が維持されます。
- 不要になったメッセージを削除しない。これにより、新しいメッセージ・カタログでも、古いプログラムがそのまま動作します。ただし、メッセージが不要になったことを示すコメントを、メッセージ・ファイルに追加できます。
- メッセージの本質的でない変更は行わない。メッセージを変更すると、通常は再翻訳が必要となるため、このコストと、変更の必要性を比較してください。一般的に、不正なパラメータ (番号やタイプ)、不正な情報、および綴りや文法の甚だしい誤りがある場合だけ、メッセージを変更します。
- ファイル内のメッセージの位置を変えずに、メッセージを訂正する。これにより、古いプログラムやカタログと、新しいプログラムやカタログ

間の不一致を避けることができます。また、訂正内容を説明するコメントを、ファイルに追加します。

### 3.5 メッセージとロケール・データの表示

メッセージ・カタログを作成した後で、意図したとおりのメッセージがカタログに含まれ、メッセージとメッセージ・セットが適切な順序になっていることを確認するために、カタログの内容を表示することができます。アプリケーションには、プログラムと同様に、実行時にロケール設定を判定し、ロケールに依存したデータを取り出し、さらに、ロケールに依存した方法でメッセージを表示するのに必要なスクリプトを含めることもできます。

以下のリストでは、メッセージ・カタログ内のメッセージを表示する `dspcat` コマンド、`dspmsg` コマンドおよび `printf` コマンドと、現在のロケール情報を表示する `locale` コマンドについて説明します。

- `dspcat` コマンド

`dspcat` コマンドは、すべてのメッセージ、特定のセット内のすべてのメッセージ、または特定のメッセージを表示できます。次の例は、`xpg4demo.cat` というカタログの 2 番目のセット内の 4 番目のメッセージを表示します。

```
% cd /usr/examples/xpg4demo/en_US
% dspcat xpg4demo.cat 2 4
Are these the changes you want to make?%
```

`dspcat` コマンドには、カタログ全体またはメッセージ・セットの出力ストリームを、パイプを通して `gencat` コマンドに渡せるように再フォーマットする `-g` フラグもあります。このオプションは、エンド・ユーザ・サイトにおけるアプリケーションの更新作業の一環として、カタログ内のメッセージ・セットを使用して、別のカタログ内のメッセージ・セットに追加や置き換えを行うときに有用です。`dspcat -g` コマンドは、既存のメッセージ・カタログからソース・ファイルを作成するときにも使用できます。ソース・ファイルが作成できたら、そのソース・ファイルをエンド・ユーザ向けに翻訳またはカスタマイズして、翻訳されたソースから新しいカタログを `gencat` コマンドで作成します。

次の例は、まず `en_US.ISO8859-1` ロケールで `du` コマンドが使用するメッセージ・カタログのメッセージ・ソースを表示して、そのソースを編集可能なファイルにリダイレクトします。

```
% dspcat -g /usr/lib/nls/msg/en_US.ISO8859-1/du.cat
```

```
$delset 1
```

```
$set 1
```

```
$quote "
```

```
1      "usage: du [-a|-s] [-klrx] [name ...]\n"
2      "du: Cannot find the current directory.\n"
3      "du: %s\n\
The specified pathname exceeded 255 bytes.\n"
4      "du: %s\n\
The generated pathname exceeded 255 bytes.\n"
5      "du: Cannot change directory to ../%s \n"
6      "Out of memory"
```

```
% dspcat -g \
/usr/lib/nls/msg/en_US.ISO8859-1/du.cat > du.msg
```

- dspmsg コマンド

dspmsg コマンドは、カタログ内の特定のメッセージを表示します。また、オプションとしてメッセージ中のすべての %s や %n \$s の指示子をテキスト文字列で置き換えることもできます。次に例を示します。

```
% dspmsg xpg4demo.cat -s 1 9 'Cannot open %s for output' xpg4demo.dat
Cannot open xpg4demo.dat for output%
```

- printf コマンド

printf コマンドは、書式付けされた文字列を標準出力に書き込みます。このコマンドは printf() 関数と同様に、ロケールに従ってメッセージを書式付けできる変換指定子をサポートしています。このコマンドを locale コマンドとともにスクリプト内で使用して、ユーザの母国語での「肯定および否定」の応答を処理することもできます。次に例を示します。

```
if printf "%s\n" "$response" | grep -Eq "`locale yesexpr`"
then
    <processing for an affirmative response goes here>
else
    <processing for a response other than affirmative goes here>
fi
```

- locale コマンド

locale コマンドは、現在のロケール設定を表示したり、どのようなロケールがシステムにインストールされているかを表示します。次の例では、locale コマンドはまず、すべてのロケール変数の現在の設定値を表示し、次に特定の変数 (LC\_MESSAGES) のキーワードと値を表示して、最後にロケール・データの特定の項目の値を表示します。

```
% locale
LANG=en_US.ISO8859-1
LC_COLLATE="en_US.ISO8859-1"
LC_CTYPE="en_US.ISO8859-1"
LC_MONETARY="en_US.ISO8859-1"
LC_NUMERIC="en_US.ISO8859-1"
LC_TIME="en_US.ISO8859-1"
LC_MESSAGES="en_US.ISO8859-1"
LC_ALL=
% locale -ck LC_MESSAGES
LC_MESSAGES
yesexpr="^([yY]|[yY][eE][sS])"
noexpr="^([nN]|[nN][oO])"
yesstr="yes:y:Y"
nostr="no:n:N"
% locale yesexpr
^([yY]|[yY][eE][sS])
```

上記のコマンドについては、`dspcat(1)`、`dspmsg(1)`、`printf(1)`、および `locale(1)` のリファレンス・ページを参照してください。

## 3.6 プログラムからのメッセージ・カタログへのアクセス

プログラムは、メッセージ・カタログを参照するために、次の関数を呼び出します。

- `catopen()` - メッセージ・カタログをオープンする (3.6.1 項)
- `catclose()` - メッセージ・カタログをクローズする (3.6.2 項)
- `catgets()` - プログラム・メッセージを読み取る (3.6.3 項)

メッセージ・カタログは通常、`NLSPATH` 環境変数の設定に従って検索されます。以降の項では、この変数と、上記の関数呼び出しについて説明します。

### 3.6.1 メッセージ・カタログのオープン

プログラムは `catopen()` 関数を呼び出して、メッセージ・カタログをオープンします。次に例を示します。

```
#include <locale.h>
#include <nl_types.h>
:
:
nl_catd          MsgCat;
:
:
```



```
setlocale(LC_ALL, "");  
:  
:  
MsgCat = catopen("new_application.cat", NL_CAT_LOCALE);
```

この例では、`catopen()` 関数はメッセージ・カタログ記述子を `MsgCat` 変数に返します。記述子が格納されるこの変数は、`nl_catd` 型として宣言されています。`catopen()` 関数と `nl_catd` 型は、ヘッダ・ファイル `/usr/include/nl_types.h` で定義されています。プログラムでは、このヘッダ・ファイルをインクルードしなければなりません。`catopen()` の呼び出しには、次の引数が必要です。

- カタログの名前

一般に、カタログ名はディレクトリ・パスを前に付けず、`filename.cat` (あるいは、値が `filename.cat` であるプログラム変数) として指定します。`catopen()` 関数は実行時に、`NLSPATH` 環境変数で定義されているパス名フォーマットに名前引数を取り込んで、カタログの完全パス名を作成します。カタログ名の引数内にスラッシュ (/) を指定した場合、`catopen()` 関数は、指定されたカタログ名を完全パス名と見なし、実行時に `NLSPATH` 変数の値を参照しません。

- `oflag` 引数

この引数は、`NL_CAT_LOCALE` 定数 (`/usr/include/nl_types.h` で定義される) またはゼロ (0) です。`NL_CAT_LOCALE` 定数が指定されている場合、`catopen()` は、`LC_MESSAGES` 環境変数のロケール・セットをサポートするメッセージ・カタログを検索します。0 を指定した場合には、`catopen()` 関数は `LANG` 環境変数のロケール・セットをサポートするメッセージ・カタログを検索します。

引数 0 は、XPG3 との互換性のためにサポートされています。

`NL_CAT_LOCALE` 引数は The Open Group の最新の UNIX CAE Specifications に準拠しているため、この引数を使用することをお勧めします。

`LC_MESSAGES` の設定値は通常、明示的に設定されるのではなく、`LANG` 設定から継承されます。ただし、プログラムまたはユーザが、`LANG` の設定とは異なるロケールを `LC_MESSAGES` に設定することもあります。

メッセージ・カタログの名前とインストール位置は、システムによって異なります。このため、The Open Group の UNIX 標準では、プログラムが

実行されるシステム上でメッセージ・カタログの検索パスとパス名の形式を定義するための NLSPATH 環境変数を規定しています。catopen() 関数は、プログラムがオープンしたカタログを見つけるために、実行時にこの変数の設定値を参照します。アプリケーションのメッセージ・カタログをユーザ・システム上の通常的位置にインストールしない場合、アプリケーションのスタートアップ手順で、NLSPATH の現在の検索パスの前に適切なパス名を付加する必要があります。

NLSPATH 環境変数に値を設定する構文は、次のとおりです。

**NLSPATH=** [ [[:]] [/directory] [[/] | [substitution-field] | [literal]] ...  
[[:] alternate\_pathname] ...]

先頭にあるコロン (:) と 2 つ連続するコロン (::) は、現在のディレクトリを示します。これ以外のコロンは、パス名の区切り文字として扱われます。検索パス内の各パス名は、次の構成要素から組み立てられます。

- カタログ・ディレクトリの完全パスを示す /directory。  
相対パスを示すときは、./directory の形式で指定できます。
- substitution-field。次のディレクティブのいずれかを指定します。

– %N

catopen() の第 1 引数の値。たとえば、次の呼び出しでは xpg4demo.cat です。

```
catopen("xpg4demo.cat", NL_CAT_LOCALE);
```

– %L

次のいずれかのロケール・セット。

catopen() の第 2 引数が NL\_CAT\_LOCALE 定数の場合は、LC\_MESSAGES。

catopen() の第 2 引数がゼロ (0) の場合は、LANG。

この置換フィールドは、fr\_FR.ISO8859-1 のように、ロケール名全体を表します。

– %l

LC\_MESSAGES 変数または LANG 変数のロケール・セット (%L と同じ条件で決定) の言語部分。

fr\_FR.ISO8859-1 というロケール名では、この置換フィールドは fr の部分を表します。

– %t

LC\_MESSAGES 変数または LANG 変数のロケール・セット (%L と同じ条件で決定) の地域部分。

fr\_FR.ISO8859-1 というロケール名では、この置換フィールドは FR の部分を表します。

– %c

LC\_MESSAGES 変数または LANG 変数のロケール・セット (%L と同じ条件で決定) のコードセット部分。

fr\_FR.ISO8859-1 というロケール名では、この置換フィールドは ISO8859-1 の部分を表します。

– %%

1 つの % 文字を表す。

• 次の項目を示す *literal*。

- 置換フィールドを使用して指定することができないディレクトリ名またはファイル名。
- 言語、地域、およびコードセットの置換フィールド間の下線 ( ) やピリオド (.), または %L 置換フィールドと %N 置換フィールド間のスラッシュ (/) などのフィールド区切り文字。

LC\_MESSAGES 設定、NLSPATH 設定、および catopen() 関数の相互関係を明確にするために、次のような条件を考えてみます。

- LC\_MESSAGES のロケール・セットは、fr\_FR.ISO8859-1 である。  
(ユーザまたはプログラムによって明示的に設定されていないかぎり、LC\_MESSAGES のロケール・セットは、LANG のロケール・セットから取り込まれます。)

- NLSPATH 変数には、次の値が設定される。

```
:%l_%t/%N:/usr/kits/xpg4demo/msg/%l_%t/%N:\n/usr/lib/nls/msg/%L/%N
```

- プログラムは、次の呼び出しでロケールを初期化する。

```
:\nsetlocale(LC_ALL, "");
```

⋮

- プログラムは、次の呼び出しでメッセージ・カタログをオープンする。

⋮

```
MsgCat = catopen("xpg4demo.cat", NL_CAT_LOCALE);
```

⋮

上記のような条件では、`catopen()` 関数は、実行時に次に示すパス名の順にカタログを検索します。

1. `xpg4demo.cat`
2. `./fr_FR/xpg4demo.cat`
3. `/usr/kits/xpg4demo/msg/fr_FR/xpg4demo.cat`
4. `/usr/lib/nls/msg/fr_FR.ISO8859-1/xpg4demo.cat`

実行時に発生した問題の原因を調べるときは、設定されていない変数があつた場合の `catopen()` の動作を考察してみます。

`LC_MESSAGES` が (直接または `LANG` 変数を通して) 設定されていない場合、`%L` および `%l` のフィールドは値 `C` (`LC_MESSAGES` のデフォルト・ロケール) となり、`%t` および `%c` の置換フィールドは検索パスから削除されます。この場合、`catopen()` は次の順序でカタログを検索します。

1. `xpg4demo.cat`
2. `./C_/xpg4demo.cat`
3. `/usr/kits/xpg4demo/msg/C/xpg4demo.cat`
4. `/usr/lib/nls/msg/C/xpg4demo.cat`

`LC_MESSAGES` は設定されているが、`NLSPATH` 変数が設定されていない場合、`catopen()` 関数は、ペンダが定義する省略時の検索パスを使用してカタログを検索します。Tru64 UNIX システムでは、省略時の検索パスは `/usr/lib/nls/msg/%L/%N:` です。この例では、省略時の設定により、`catopen()` の検索順序は次のようになります。

1. `/usr/lib/nls/msg/fr_FR.ISO8859-1/xpg4demo.cat`
2. `xpg4demo.cat`

最後に、LC\_MESSAGES と NLSPATH のどちらも設定されていない場合、catopen() は次の順序で検索を実行します。

1. /usr/lib/nls/msg/xpg4demo.cat
2. ./xpg4demo.cat

該当するロケールのメッセージ・カタログが見つからなかった場合、catopen() 関数は次に /usr/share/.msg\_conv-locale-name ファイルが存在するかどうか確認します。このファイルは、メッセージ・カタログが用意されている別のロケールを指定するためのファイルです。このファイルが見つかると、利用できるメッセージ・カタログがオープンされ、適切なコードセット変換コンバータが起動され、LC\_MESSAGES に設定されているコードセットにメッセージが変換されます。たとえば、ISO8859-1 フォーマットのフランス語の catalog\_name が存在し、.msg\_conv-fr\_FR.UTF-8 ファイルでそれを指定している場合、このカタログがオープンされ UTF-8 フォーマットにメッセージが変換されます。

catopen() 関数は、指定されたメッセージ・カタログがオープンできない場合、エラー・ステータスは返しません。プログラムの性能を向上するために、カタログを参照する最初の catgets() 呼び出しが実行されるまで、カタログは実際にはオープンされません。catopen() の呼び出し時に、ファイルのオープンに失敗したことをプログラムで検出する必要がある場合は、catopen() を呼び出した直後に catgets() を呼び出さなければなりません。これにより、catgets() の呼び出しでエラーが返されたときに、プログラムを終了させることができます。メッセージ・カタログからメッセージを取り出す前に多くの処理を実行するプログラムでは、早い段階で catgets() を呼び出すようにします。ただし、カタログのオープンに成功しないかぎり、ユーザの母国語に翻訳されたエラー・メッセージを取り出すことは不可能なため、この特別なエラーをユーザに通知することには問題があります。

catopen() 関数についての詳細は、catopen(3) を参照してください。

---

#### 注意

実効ユーザ ID が root に設定されているプロセスから呼び出された場合、catopen() 関数は NLSPATH の設定を無視し、パス /usr/lib/nls/msg/%L/%N を使用してメッセージ・カタログを検索します。プログラムが実効ユーザ ID を root に設定し

て実行している場合は、次の処理のいずれかを実行しなければなりません。

- そのプログラムが使用するすべてのメッセージ・カタログを、`/usr/lib/nls/msg/%L` で示されるロケール・ディレクトリにインストールする。
- または、そのプログラムが使用するメッセージ・カタログを他のディレクトリにインストールし、`/usr/lib/nls/msg/%L` ディレクトリに、それらのカタログ・ファイルへのリンクを作成する。

この制約は、スーパーユーザとしてログインしているユーザが起動するプログラムには適用されません。この制約は、`setuid()` 呼び出しを実行して、実効ユーザ ID が `root` になるようなサブプロセスを生成するプログラムにのみ適用されます。

---

### 3.6.2 メッセージ・カタログのクローズ

`catclose()` 関数は、メッセージ・カタログをクローズします。この関数には引数が 1 つあり、`catopen()` 関数から返されたカタログ記述子を指定します。次に例を示します。

```
(void) catclose(MsgCat);
```

`exit()` 関数もまた、プロセスの終了時にオープンされているメッセージ・カタログをクローズします。

### 3.6.3 プログラム・メッセージの読み取り

`catgets()` 関数は、メッセージをプログラムに読み込みます。この関数は次の引数を取ります。

- `catopen()` 呼び出しで返されたメッセージ・カタログ記述子
- メッセージ・セットのシンボリック識別子または数値識別子  
ユーザ定義のメッセージ・セットを含まないメッセージ・カタログからメッセージを取り出すときは、`NL_SETD` 定数を指定します。
- メッセージのシンボリック識別子または数値識別子
- デフォルト・メッセージ文字列

プログラムは、指定したメッセージをカタログから取り出せなかったときに、このデフォルト・メッセージ文字列を使用します。このような状況は、カテゴリが見つからない場合やカテゴリをオープンできない場合に発生します。プログラムではデフォルト文字列を共通に使用するため、デフォルト・テキストは、意味があり、必ず利用できるものでなければなりません。

一般に、`catgets()` 関数は他のルーチンと組み合わせて使用します。その場合、この関数は直接呼び出すこともできますが、プログラム定義のマクロとして呼び出すこともできます。次に示すサンプル・プログラム `xpg4demo` のコードでは、特定のメッセージ・セットにアクセスするマクロを定義し、そのマクロを `printf()` ルーチンの引数として使用します。

```
...
#define GetMsg(id, defmsg)\
    catgets(MsgCat, MSGInfo, id, defmsg)
...

printf(GetMsg(I_COM_DISP_LIST_FMT,
    "%6ld %20S %-30S %3S %10s\n"),
    emp->badge_num,
    emp->first_name,
    emp->surname,
    emp->cost_center,
    buf);
...
```

詳細については、`catgets(3)` のリファレンス・ページを参照してください。

---

#### 注意

`gettext()` 関数も、メッセージ・カタログからメッセージを読み取ります。この関数は System V Interface Definition (SVID) には含まれていますが、X/Open の UNIX 標準では規定されていません。この関数については、`gettext(3)` のリファレンス・ページを参照してください。

---





---

## curses ライブラリ・ルーチンを使用した ワイド文字の処理

curses ライブラリには、文字端末用のユーザ・インタフェースを開発するための関数が含まれています。この章では、curses ライブラリ関数の拡張機能について説明します。これらの拡張機能により、マルチバイト文字を収納できるワイド文字フォーマットが使用できるようになります。

ワイド文字や複合文字フォーマットのマルチバイト文字の処理には、X/Open Curses CAE Specifications Version 4.2 に準拠する関数を使用してください。これらの関数は、System V 国際化機能 (MNLS: *Multi-National Language Supplement*) に規定されている関数に代わるものです。

この章では、画面上の、またはキーボードから入力した文字や文字列を処理する curses 関数とマクロの概要について説明します。各節の表では、同じ処理を実行できる curses インタフェースが複数あるかどうかと、国際化ソフトウェアの作成に最も適した推奨 curses インタフェースを明記します。つまり、この表では、ワイド文字や複合文字フォーマットを扱い、X/Open Curses CAE 仕様に準拠している curses ライブラリ関数やマクロに重点を置きます。アプリケーションでは、表中の「推奨ルーチン」欄にリストされている curses インタフェースを使用してください。

各インタフェースの構文と詳細については、セクション 3 のリファレンス・ページを参照してください。この章は、実行したい操作に必要なインタフェースを見つけるためにご利用ください。その後、man コマンドを使用して、必要なインタフェースのリファレンス・ページを表示してください。curses ライブラリに含まれる関数の概要については、curses(3) のリファレンス・ページを参照してください。

---

### 注意

---

一部の curses ルーチンは、curses ウィンドウ上に表示されている文字を上書きします。文字を完全に上書きできるルーチンは、wchar\_t または cchar\_t 型のデータを使用するルーチンだけで

す。マルチバイト文字のように、上書きされる文字の表示幅が 1 カラムより大きいときは、これらのインタフェースは文字の残った部分を消去するために、空白文字を書き込みます。たとえば、2 カラム幅の中国語文字の 1 カラム目を英字の a で上書きした場合、中国語文字の 2 カラム目は、空白文字で上書きされます。

国際化されていない `curses` ルーチンでマルチバイト文字を上書きしたときの動作は未定義です。

## 4.1 `curses` ウィンドウへのワイド文字の書き込み

以下の項では、`curses` ウィンドウ上にワイド文字を追加したり、挿入するルーチンを説明します。ターゲットとなる場所に既に文字が存在する場合、これらのルーチンは次のいずれかの操作を行います。

- 既存の文字を上書きしカーソルを次に進める。
- 既存の文字の前に新しい文字を挿入し、カーソルは進めない。

### 4.1.1 ワイド文字を追加 (上書き) し、カーソルを進める

表 4-1 のルーチンは、画面上のウィンドウにワイド文字とその属性を追加し、カーソルを進めます。文字がすでにターゲット位置に存在するときは、その文字は追加する文字で上書きされます。

ルーチンを選択する条件は、次のとおりです。

- 省略時のウィンドウと指定したウィンドウのどちらに文字を追加するか。
- 現在の座標と指定した座標のどちらに文字を追加するか。
- 画面を再表示するかどうか。

これらのルーチンにワイド文字とその属性を渡すには、`const cchar_t` データ型を使用します。

表 4-1: ワイド文字を追加し，カーソルを進める **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
add_wch	addch , addwch	ウィンドウ: 省略時 位置: 現在 画面の再表示: しない
wadd_wch	waddch , waddwch	ウィンドウ: 指定 位置: 現在 画面の再表示: しない
mvadd_wch	mvaddch , mvaddwch	ウィンドウ: 省略時 位置: 指定 画面の再表示: しない
mvwadd_wch	mvwaddch , mvwaddwch	ウィンドウ: 指定 位置: 指定 画面の再表示: しない
echo_wchar	echowchar	ウィンドウ: 省略時 位置: 現在 画面の再表示: する
wecho_wchar	wechowchar	ウィンドウ: 指定 位置: 現在 画面の再表示: する

4.1.2 ワイド文字を (上書きせずに) 挿入し，カーソルを進めない

表 4-2のルーチンは，現在の座標または指定した座標でワイド文字列をウィンドウに挿入し，書き込み操作後にカーソル位置を変更しません。ワイド文字は，ターゲット位置にある既存の文字の前に挿入されます。このため，これらのルーチンは，行内にすでにある文字は上書きしません。ターゲット位置とその右側にある既存の文字は右方向に移動され，右端の文字は切り捨てられます。このカテゴリのインタフェースを選択する条件は，次のとおりです。

- 省略時のウィンドウと指定したウィンドウのどちらに書き込むか。
- 現在の座標と指定した座標のどちらに書き込むか。

表 4-2: ワイド文字を挿入し、カーソルを進めない **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
ins_wch	insch , inswch	ウィンドウ: 省略時 位置: 現在
wins_wch	winsch , winswch	ウィンドウ: 指定 位置: 現在
mvins_wch	mvinsch , mvinswch	ウィンドウ: 省略時 位置: 指定
mvwins_wch	mvwinsch , mvwinswch	ウィンドウ: 指定 位置: 指定

## 4.2 curses ウィンドウへのワイド文字列の書き込み

以下の項では、**curses** ウィンドウに文字列を追加したり、挿入するルーチンについて説明します。

### 4.2.1 ワイド文字列を追加 (上書き) し、カーソルを進めない

表 4-3のルーチンは、ワイド文字列とその文字属性をウィンドウに追加します。ただしこれらのルーチンには、次の特徴があります。

- カーソルの位置を進めない。
- 通常はカーソル位置に影響を与える特殊文字 (改行、タブ、後退など) が文字列に含まれているかどうかを検査しない。
- 文字列を次の行に折り返えさずに切り捨てる。

これらのルーチンでは、ターゲット位置にすでに存在している文字は、追加する文字列の文字で上書きされます。このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 文字列中のすべての文字を書き込むか、それとも一部の文字だけを書き込むか。
- 省略時のウィンドウと指定したウィンドウのどちらに文字を書き込むか。
- 現在の座標と指定した座標のどちらに文字を書き込むか。

表 4-3: ワイド文字列を追加しカーソルを進めない **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
<code>add_wchstr</code>	<code>addwchstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 現在
<code>add_wchnstr</code>	<code>addwchnstr</code>	文字数: 指定 ウィンドウ: 省略時 位置: 現在
<code>wadd_wchstr</code>	<code>waddwchstr</code>	文字数: すべて ウィンドウ: 指定 位置: 現在
<code>wadd_wchnstr</code>	<code>waddwchnstr</code>	文字数: 指定 ウィンドウ: 指定 位置: 現在
<code>mvadd_wchstr</code>	<code>mvaddwchstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 指定
<code>mvadd_wchnstr</code>	<code>mvaddwchnstr</code>	文字数: 指定 ウィンドウ: 省略時 位置: 指定
<code>mvwadd_wchstr</code>	<code>mvwaddwchstr</code>	文字数: すべて ウィンドウ: 指定 位置: 指定
<code>mvwadd_wchnstr</code>	<code>mvwaddwchnstr</code>	文字数: 指定 ウィンドウ: 指定 位置: 指定

4.2.2 ワイド文字列を追加 (上書き) し、カーソルを進める

4.2.1 項で説明したルーチンと同じように、表 4-4 のルーチンもワイド文字列 (ビデオ文字属性は除く) をウィンドウに追加し、既存の文字を上書きします。ただし、これらのルーチンは、次の処理も実行します。

- カーソル位置を進める。

- 文字位置に影響を与える特殊文字 (改行, タブ, 後退など) が文字列に含まれているかどうかを検査する。
- 文字列を切り捨てずに, 次の行に折り返す。

このカテゴリのインタフェースを選択する条件は, 次のとおりです。

- 文字列中のすべての文字を書き込むか, それとも一部の文字だけを書き込むか。
- 省略時のウィンドウと指定したウィンドウのどちらに文字を書き込むか。
- 現在の座標と指定した座標のどちらに文字を書き込むか。

表 4-4: ワイド文字列を追加しカーソルを進める **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
<code>addwstr</code>	<code>addstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 現在
<code>addnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 省略時 位置: 現在
<code>waddwstr</code>	<code>waddstr</code>	文字数: すべて ウィンドウ: 指定 位置: 現在
<code>waddnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 指定 位置: 現在
<code>mvaddwstr</code>	<code>mvaddstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 指定
<code>mvaddnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 省略時 位置: 指定

表 4-4: ワイド文字列を追加しカーソルを進める `curses` ルーチン (続き)

推奨ルーチン	代替ルーチン	動作
<code>mvwaddwstr</code>	<code>mvwaddstr</code>	文字数: すべて ウィンドウ: 指定 位置: 指定
<code>mvwaddnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 指定 位置: 指定

4.2.3 ワイド文字列を (上書きせずに) 挿入し、カーソルを進めない

表 4-5 で説明するルーチンは、`curses` ウィンドウ内のターゲット位置の直前に、ワイド文字列を挿入します。これらのルーチンには、次の特徴があります。

- ターゲット位置とその右側にある既存の文字を、さらに右方向に移動する。  
既存の文字は上書きされませんが、行の終わりで右端の文字が切り捨てられることがあります。
- 文字位置とカーソル位置に影響を与える特殊文字 (改行、タブ、後退など) が文字列に含まれているかどうかを検査する。
- 書き込み操作後にカーソルを進めない。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 文字列中のすべての文字を書き込むか、それとも一部の文字だけを書き込むか。
- 省略時のウィンドウと指定したウィンドウのどちらに文字を書き込むか。
- 現在の座標と指定した座標のどちらに文字を書き込むか。

表 4-5: ワイド文字列を挿入しカーソルを進めない `curses` ルーチン

推奨ルーチン	代替ルーチン	動作
<code>ins_wstr</code>	<code>inswstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 現在
<code>ins_nwstr</code>	<code>insnwstr</code>	文字数: 指定 ウィンドウ: 省略時 位置: 現在
<code>wins_wstr</code>	<code>winswstr</code>	文字数: すべて ウィンドウ: 指定 位置: 現在
<code>wins_nwstr</code>	<code>winsnwstr</code>	文字数: 指定 ウィンドウ: 指定 位置: 現在
<code>mvins_wstr</code>	<code>mvinswstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 指定
<code>mvins_nwstr</code>	<code>mvinsnwstr</code>	文字数: 指定 ウィンドウ: 省略時 位置: 指定
<code>mvwins_wstr</code>	<code>mvwinswstr</code>	文字数: すべて ウィンドウ: 指定 位置: 指定
<code>mvwins_nwstr</code>	<code>mvwinsnwstr</code>	文字数: 指定 ウィンドウ: 指定 位置: 指定

### 4.3 `curses` ウィンドウからのワイド文字の削除

表 4-6 のルーチンは、`curses` ウィンドウ内のターゲット位置にあるワイド文字を削除します。その行の、削除された文字以降の文字は、1 文字分左にシフトされます。これらのルーチンは、マルチバイト文字がサポートされる前から `curses` ライブラリに含まれており、ワイド文字フォーマットを正しく処理できるように再定義されています。



このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 省略時のウィンドウと指定したウィンドウのどちらの文字を削除するか。
- 現在の座標と指定した座標のどちらの文字を削除するか。

表 4-6: ワイド文字列を削除する **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
<code>delch</code>	代替ルーチンなし	ウィンドウ: 省略時 位置: 現在
<code>wdelch</code>	代替ルーチンなし	ウィンドウ: 指定 位置: 現在
<code>mvdelch</code>	代替ルーチンなし	ウィンドウ: 省略時 位置: 指定
<code>mvwdelch</code>	代替ルーチンなし	ウィンドウ: 指定 位置: 指定

## 4.4 **curses** ウィンドウからのワイド文字の読み取り

表 4-7 のルーチンは、**curses** ウィンドウからワイド文字とそのビデオ属性を読み取ります。プログラムに返されるデータは `cchar_t` 型なので、ワイド文字と属性の両方が格納されます。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 省略時のウィンドウと指定したウィンドウのどちらから文字を読み取るか。
- 現在の座標と指定した座標のどちらから文字を読み取るか。

表 4-7: ウィンドウからワイド文字を読み取る **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
<code>in_wch</code>	<code>inch</code> , <code>inwch</code>	ウィンドウ: 省略時 位置: 現在
<code>win_wch</code>	<code>winch</code> , <code>winwch</code>	ウィンドウ: 指定 位置: 現在

表 4-7: ウィンドウからワイド文字を読み取る `curses` ルーチン (続き)

推奨ルーチン	代替ルーチン	動作
<code>mvin_wch</code>	<code>mvinch</code> , <code>mvinwch</code>	ウィンドウ: 省略時 位置: 指定
<code>mvwin_wch</code>	<code>mvwinch</code> , <code>mvwinwch</code>	ウィンドウ: 指定 位置: 指定

## 4.5 `curses` ウィンドウからのワイド文字列の読み取り

`curses` ウィンドウからワイド文字列を読み取るルーチンには、2つのセットがあります。4.5.1項の一連のルーチンは、ビデオ属性付きのワイド文字の文字列を読み取ります。4.5.2項の一連のルーチンは、ワイド文字の文字列を、属性を取り除いて読み取ります。

### 4.5.1 属性付きのワイド文字列の読み取り

表 4-8 のルーチンは、`curses` ウィンドウからワイド文字列とその文字属性を読み取ります。推奨ルーチンから返される文字列は、`cchar_t` データ型です。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 文字列中のすべてのワイド文字を読み取るか、それとも指定した数のワイド文字を読み取るか。
- 省略時のウィンドウと指定したウィンドウのどちらから文字を読み取るか。
- 現在の座標と指定した座標のどちらから文字を読み取るか。

表 4-8: ワイド文字列とその属性を読み取る `curses` ルーチン

推奨ルーチン	代替ルーチン	動作
<code>in_wchstr</code>	<code>inwchstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 現在
<code>in_wchnstr</code>	<code>inwchnstr</code>	文字数: 指定 ウィンドウ: 省略時 位置: 現在

表 4-8: ワイド文字列とその属性を読み取る **curses** ルーチン (続き)

推奨ルーチン	代替ルーチン	動作
<code>win_wchstr</code>	<code>winwchstr</code>	文字数: すべて ウィンドウ: 指定 位置: 現在
<code>win_wchnstr</code>	<code>winwchnstr</code>	文字数: 指定 ウィンドウ: 指定 位置: 現在
<code>mvin_wchstr</code>	<code>mvinwchstr</code>	文字数: すべて ウィンドウ: 省略時 位置: 指定
<code>mvin_wchnstr</code>	<code>mvinwchnstr</code>	文字数: 指定 ウィンドウ: 省略時 位置: 指定
<code>mvwin_wchstr</code>	<code>mvwinwchstr</code>	文字数: すべて ウィンドウ: 指定 位置: 指定
<code>mvwin_wchnstr</code>	<code>mvwinwchnstr</code>	文字数: 指定 ウィンドウ: 指定 位置: 指定

4.5.2 属性なしでのワイド文字列の読み取り

表 4-9 のルーチンは、`curses` ウィンドウからワイド文字列を読み取り、`wchar_t` データ型の文字列をプログラムの変数に格納します。ビデオ属性は、文字列に格納される文字から取り除かれます。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 文字列中のすべてのワイド文字を読み取るか、それとも指定した数のワイド文字を読み取るか。
- 省略時のウィンドウと指定したウィンドウのどちらから文字を読み取るか。
- ウィンドウ内の現在の座標と指定した座標のどちらから文字を読み取るか。

表 4-9: 属性なしでワイド文字列を読み取る **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
<code>inwstr</code>	代替ルーチンなし	文字数: すべて ウィンドウ: 省略時 位置: 現在
<code>innwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 省略時 位置: 現在
<code>winwstr</code>	代替ルーチンなし	文字数: すべて ウィンドウ: 指定 位置: 現在
<code>winnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 指定 位置: 現在
<code>mvinwstr</code>	代替ルーチンなし	文字数: すべて ウィンドウ: 省略時 位置: 指定
<code>mvinnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 省略時 位置: 指定
<code>mvwinwstr</code>	代替ルーチンなし	文字数: すべて ウィンドウ: 指定 位置: 指定
<code>mvwinnwstr</code>	代替ルーチンなし	文字数: 指定 ウィンドウ: 指定 位置: 指定

## 4.6 端末からの文字列の読み取り

表 4-10 のルーチンは、`curses` ウィンドウに対応する端末から文字列を読み取り、プログラムのバッファに格納します。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

### 4-12 `curses` ライブラリ・ルーチンを使用したワイド文字の処理

- 文字列中のすべてのワイド文字を読み取るか、それとも指定した数のワイド文字を読み取るか。
- 読み取った文字を、省略時のウィンドウと指定したウィンドウのどちらで使用するか。
- 読み取った文字を、ウィンドウ内の現在の座標と指定した座標のどちらで使用するか。

表 4-10: ワイド文字列を端末から読み取るcursesルーチン

推奨ルーチン	代替ルーチン	動作
get_wstr	getstr , getwstr	文字数: すべて ウィンドウ: 省略時 位置: 現在
getn_wstr	getnwstr	文字数: 指定 ウィンドウ: 省略時 位置: 現在
wget_wstr	wgetstr , wgetwstr	文字数: すべて ウィンドウ: 指定 位置: 現在
wgetn_wstr	wgetnwstr	文字数: 指定 ウィンドウ: 指定 位置: 現在
mvget_wstr	mvgetstr , mvgetwstr	文字数: すべて ウィンドウ: 省略時 位置: 指定
mvgetn_wstr	mvgetnwstr	文字数: 指定 ウィンドウ: 省略時 位置: 指定
mvwget_wstr	mvwgetstr , mvwgetwstr	文字数: すべて ウィンドウ: 指定 位置: 指定
mvwgetn_wstr	mvwgetnwstr	文字数: 指定 ウィンドウ: 指定 位置: 指定

## 4.7 キーボードからのワイド文字の読み取りまたはキューイング

表 4-11 のルーチンは、`curses` ウィンドウに対応する端末のキーボードからシングルバイト文字またはマルチバイト文字を読み取り、その文字をワイド文字フォーマットに変換して、プログラムに返します。`curses` の入力モードが `noecho` に設定されていない場合、これらのルーチンは、すべての文字を画面上に表示します。

`unget_wch` インタフェースはワイド文字を入力キューの先頭に置きます。この場合、次の `wget_wch` 呼び出しにより、入力キューの文字がプログラムに返されます。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 読み込んだ文字を、省略時のウィンドウと指定したウィンドウのどちらで使用するか。
- 読み込んだ文字を、ウィンドウ内の現在の位置と指定した位置のどちらで使用するか。
- ただちに使用するか、それとも後で使用するか。

表 4-11: ワイド文字列をキーボードから読み取る **curses** ルーチン

推奨ルーチン	代替ルーチン	動作
<code>get_wch</code>	<code>getch</code> , <code>getwch</code>	ウィンドウ: 省略時 位置: 現在
<code>wget_wch</code>	<code>wgetch</code> , <code>wgetwch</code>	ウィンドウ: 指定 位置: 現在
<code>mvget_wch</code>	<code>mvgetch</code> , <code>mvgetwch</code>	ウィンドウ: 省略時 位置: 指定
<code>mvwget_wch</code>	<code>mvwgetch</code> , <code>mvwgetwch</code>	ウィンドウ: 指定 位置: 指定
<code>unget_wch</code>	<code>ungetch</code> , <code>ungetwch</code>	ウィンドウ: 適用外 位置: 適用外 入力キュー: 文字をキュー付け

## 4.8 curses ウィンドウでの書式付きテキストの変換

表 4-12 のルーチンは、`curses` ウィンドウからワイド文字を読み取って変換します。これらの関数は、`curses` ライブラリが国際化される前から `curses` ライブラリに含まれており、ワイド文字データを扱えるように拡張されています。これらの関数はすべて、`wgetstr` を呼び出してウィンドウからワイド文字列を読み取った後、`scanf()` 関数の規則に従って文字を解釈して変換します。詳細については、`scanf(3)` のリファレンス・ページを参照してください。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 省略時のウィンドウと指定したウィンドウのどちらの文字列を変換するか。
- 現在の座標と指定した座標のどちらの文字列を変換するか。
- 呼び出しの引数の 1 つとして、変数リストを含める必要があるかどうか。

表 4-12: ウィンドウ内の書式付きテキストを変換する `curses` ルーチン

推奨ルーチン	代替ルーチン	動作
<code>scanw</code>	代替ルーチンなし	ウィンドウ: 省略時 位置: 現在 引数の数: 固定
<code>wscanw</code>	代替ルーチンなし	ウィンドウ: 指定 位置: 現在 引数の数: 固定
<code>mvscanw</code>	代替ルーチンなし	ウィンドウ: 省略時 位置: 指定 引数の数: 固定
<code>mvwscanw</code>	代替ルーチンなし	ウィンドウ: 指定 位置: 指定 引数の数: 固定
<code>vw_scanw</code>	<code>wscanw</code>	ウィンドウ: 指定 位置: 現在 引数の数: 可変

## 4.9 curses ウィンドウでの書式付きテキストの表示

表 4-13 のルーチンは文字列を書式付けして、`curses` ウィンドウに表示します。これらの関数は `curses` ライブラリが国際化される前から `curses` ライブラリに含まれており、ワイド文字フォーマットのデータを扱えるように再定義されています。これらの関数は、`printf()` (または `vprintf()`) と同様な形式で文字列を書式付けし、`addstr()` (または `waddstr()`) と同様な形式で文字列を書き込みます。書式付けについては、`printf(3)` のリファレンス・ページを参照してください。

このカテゴリのインタフェースを選択する条件は、次のとおりです。

- 省略時のウィンドウと指定したウィンドウのどちらに表示するか。
- 現在の位置と指定した位置のどちらに表示するか。
- 呼び出しの引数の 1 つとして、変数リストを含める必要があるかどうか。

表 4-13: ウィンドウに書式付きテキストを表示する `curses` ルーチン

推奨ルーチン	代替ルーチン	動作
<code>printw</code>	代替ルーチンなし	ウィンドウ: 省略時 位置: 現在 引数の数: 固定
<code>wprintw</code>	代替ルーチンなし	ウィンドウ: 指定 位置: 現在 引数の数: 固定
<code>mvprintw</code>	代替ルーチンなし	ウィンドウ: 省略時 位置: 指定 引数の数: 固定
<code>mvwprintw</code>	代替ルーチンなし	ウィンドウ: 指定 位置: 指定 引数の数: 固定
<code>vw_printw</code>	<code>vwprintw</code>	ウィンドウ: 指定 位置: 現在 引数の数: 可変



## 国際化対応の X , Xt , および Motif アプリケーションの作成

この章では、グラフィカル・ユーザ・インタフェースを作成するときに利用できる国際化機能について説明します。具体的には、次の各コンポーネントについて解説します。

- `libXt`。X ウィンドウ・システムのリリース 6 で利用できるツールキット・イントリンシクス・ライブラリ (5.1 節)。
- `libXm` と `libDXm`。OSF/Motif のバージョン 1.2 で利用できるライブラリと、OSF/Motif ツールキットに対する DECwindows 拡張として提供されている機能 (5.2 節)。
- `libX11`。X ウィンドウ・システムのリリース 6 で利用できる X ライブラリ (5.3 節)。

この章では、読者が X ウィンドウ・システムや OSF/Motif ツールキットをすでに理解していることを前提としています。これらのコンポーネントの詳細については、次のドキュメントを参照してください。

- 『*X Window System Environment*』。オペレーティング・システムのオンライン・ドキュメント・セットとして入手できます。
- 『*Programmer's Supplement for Release 6 of the X Window System*』。このドキュメントは、書籍として、O'Reilly and Associates, Inc. から出版されています。

詳細については、これらのドキュメントの他に、各関数のリファレンス・ページも参照してください。

この章では、共通デスクトップ環境 (CDE) に特有の国際化機能については説明しません。これらの機能の使用方法については、『*Common Desktop Environment: プログラマーズ・ガイド (国際化対応編)*』(オペレーティング・システムのオンライン・ドキュメント・セットとして入手可能) を参照してください。(オペレーティング・システムのドキュメントは、Web

サイト <http://tru64unix.compaq.co.jp/document/index.html> で参照できます。)

アプリケーションを地域化する際に便利のように、CDE 環境でロケールを直接サポートできるようにするためのファイルを、以下のリストに示します。

#### メッセージ・カタログ

`/usr/lib/nls/msg/locale_name/filename.cat...`

#### リファレンス・ページ

`/usr/share/locale_name/man/mann/refpage.n...`

#### リソース・ファイル

`/usr/lib/X11/locale_name/app-defaults/file_name`

`/usr/dt/app-defaults/locale_name/file_name`

#### UID ファイル

`/usr/lib/X11/locale_name/uid/file_name`

#### CDE アクション/データタイプ・ファイル

`/usr/dt/appconfig/types/locale_name/file_name`

#### CDE ヘルプ・ファイル

`/usr/dt/appconfig/help/locale_name/file_name`

#### CDE スタイルマネージャのバックドロップ・ファイル

`/usr/dt/backdrops/desc.locale_name`

#### CDE スタイルマネージャのパレット・ファイル

`/usr/dt/palettes/desc.locale_name`

オペレーティング・システムがサポートしているロケールには、直接サポートしているものと、間接的に (自動コードセット変換を通して) サポートしているものがあります。たとえば、オペレーティング・システムは、コードセット変換を通して `ja_JP.UTF-8` ロケールをサポートしているので、オペレーティング・システム・ソフトウェアをインストールした

後に、`locale-name` ディレクトリまたは `locale-name` 拡張子として `ja_JP.UTF-8` を含むパスを見つけることはできません。

ただし、適切にローカライズしてエンコードしたファイルを作成し、上記のリストで指定された位置にそのファイルをインストールすると、`ja_JP.UTF-8` を直接サポートできるようになります。新しいロケールを最初から作成する方法については、第 6 章を参照してください。

## 5.1 Xt イントリンシクス・ライブラリの国際化機能の使用

X ツールキット (Xt) イントリンシクス・ライブラリは、初期化処理とリソース管理に関連する国際化機能を提供します。以降の項では、これらの機能について説明します。アプリケーションで Xt イントリンシクス・ライブラリ (`libXt`) のルーチンを使用する方法については、各コンポーネントのリファレンス・ページを参照してください。

### 5.1.1 Xt 関数によるロケールの設定

国際化 Xt を使用するアプリケーションでは、ロケールに応じてリソースを解析しなければなりません。そのため、アプリケーションはリソース・データベースを初期化する前に、ロケールを設定しなければなりません。ところがリソースでもアプリケーションのロケールを設定することができます。この矛盾を解決するために、X ツールキットのリリース 5 では言語プロシージャが取り入れられました。言語プロシージャは、X ツールキットを初期化する前に登録され、ロケールを設定するのに都合のよい時点で初期化中に呼び出されます。

`XtSetLanguageProc()` 関数は、ロケールを設定するための言語プロシージャを登録します。省略時の動作では、この関数はまず、標準 C ライブラリの `setlocale()` 関数を呼び出してロケールを設定し、次に X ライブラリ関数の `XSupportsLocale()` と `XSetLocaleModifiers()` を呼び出してロケールを初期化します。

Xt のルーチンを使用するアプリケーションでは、システムの省略時の言語プロシージャを使用している場合でも、`XtSetLanguageProc()` を呼び出さなければなりません。この関数を呼び出さないと、ロケールは設定されず、他の Xt ルーチンはロケールに従った動作を行いません。

アプリケーションでロケールを設定する最も一般的な方法は、`XtAppInitialize()` を呼び出す前に、次の関数を呼び出すことです。

```
XtSetLanguageProc(NULL, NULL, NULL);
```

`XtSetLanguageProc()` を呼び出した後、アプリケーションは次の Xt 初期化関数のいずれかを呼び出せます。

- `XtInitialize()`
- `XtAppInitialize()`
- `XtOpenDisplay()`

これらの関数は、コマンド行と `RESOURCE_MANAGER` プロパティを解析することにより `xnlLanguage` リソースの値を取得する、`XtDisplayInitialize()` 関数を呼び出します。次に `XtDisplayInitialize()` は、`xnlLanguage` 値を引数として、`XtSetLanguageProc()` の呼び出しで登録されている言語プロシージャを呼び出します。この後、`XtDisplayInitialize()` は、言語プロシージャから返されたロケールのリソースを解析します。

### 5.1.2 Xt 関数におけるフォントセット・リソースの使用

Xt ルーチンは、母国語のテキストを表示するすべての国際化ウィジェットで、`XFontStruct` 構造体の代わりに `XFontSet` 構造体をサポートします。`XFontSet` をサポートするために、次のリソース属性が用意されています。

- `XtNFontSet` (リソース名)
- `XtCFontSet` (リソース・クラス)
- `XtRFontSet` (リソース表現タイプ)

X ツールキットには、`-----R-----120-75-75-----` のようなあらかじめ登録されている文字列を、構造体 (`XFontSet`) 内のフォントセットのリストに変換するコンバータが含まれています。このような文字列を適切なフォントセットに変換できなかった場合でも、このコンバータは省略時のフォントセット・リストを設定し、適切なフォントセットの代替えとして使用できるようにします。

### 5.1.3 Xt ライブラリ関数におけるテキスト入力イベントのフィルタリング

X ツールキット・イントリンシクス・ライブラリのリリース 5 以降、`XtDispatchEvent()` 関数は `XFilterEvent()` の呼び出しに置き換えられました。この変更により、入力サーバは、Xt ルーチンを使用するアプ

リケーションが登録された X イベントを処理する前に、それらのイベントを横取りできます。

#### 5.1.4 Xt ライブラリ関数におけるロケールのコードセット・コンポーネントの取り込み

X ツールキット・イントリンシクス・ライブラリのリリース 5 以降、ロケール・エンティティは、以前のリリースでサポートされていた言語コンポーネントと地域コンポーネントに加え、コードセット・コンポーネントもサポートします。

### 5.2 OSF/Motif および DECwindows Motif ツールキットの国際化機能の使用

『*Common Desktop Environment: プログラマーズ・ガイド*（国際化対応編）』では、国際化 Motif アプリケーションを開発するためのガイドラインを説明しています。このマニュアルは、オペレーティング・システムのドキュメント・セットの一部で、Web サイト <http://tru64unix.compaq.co.jp/document/index.html> で参照できます。

以降の項は、『*Common Desktop Environment: プログラマーズ・ガイド*（国際化対応編）』の内容を補足するためのものです。

#### 5.2.1 Motif アプリケーションにおける言語の設定

OSF/Motif ツールキット (libXm)、および OSF/Motif ツールキットに対する DECwindows 拡張 (libDXm) の大半の国際化機能は、X ライブラリ (libX) と X ツールキット (libXt) のリリース 5 以降で取り入れられた機能を通じてサポートされます。

Motif の国際化機能は X ライブラリと X ツールキットのリリース 6 またはリリース 6.3 をインストールした場合でも、同じようにサポートされます。たとえば、Motif アプリケーションのロケールを設定するには、Xt アプリケーションで説明したのと同じ関数セットとガイドラインを使用します (5.1.1 項を参照)。アプリケーションが、X ツールキットを初期化する前に言語プロシージャを登録するための `XtSetLanguageProc()` 呼び出しに失敗した場合、Motif ウィジェットは以降の項で説明する国際化機能をサポートしません。つまり、ウィジェットは X ツールキットのリリース 5 および OSF/Motif リリース 1.2 より前のリリースと同じ動作をします。

アプリケーションの言語は、次のいずれかの方法で指定できます。このリストは、優先度の高い順に並んでいます。たとえば、`argv` 引数による言語設定は、`RESOURCE_MANAGER` での言語設定よりも優先されます。

1. `XtAppInitialize()`、`XtOpenDisplay()`、`XtDisplayInitialize()`、または `XtOpenApplication()` の呼び出しの `argv` 引数の値
2. 指定されたディスプレイのルート・ウィンドウの `RESOURCE_MANAGER` プロパティ中の言語リソース設定
3. ユーザの `.Xdefaults` ファイル内の `xnlLanguage` リソース設定
4. `LANG` 環境変数の設定

次の点に注意してください。

- アプリケーションが最初のディスプレイをオープンすると、Motif ルーチンはアプリケーションが終了するまで、確立された言語設定を使用する。
- `RESOURCE_MANAGER` プロパティがルート・ウィンドウに設定されている場合、言語リソースが `RESOURCE_MANAGER` プロパティ中で定義されていないなくても、Motif ルーチンは `.Xdefaults` ファイルを使用しない。

## 5.2.2 コンパウンド・ストリングと、`XmText` および `XmTextField` ウィジェットの使用

OSF/Motif の `XmText` ウィジェットと `XmTextField` ウィジェットには、`X` と `X ツールキット` のライブラリをベースとした国際化機能が備わっています。これらのウィジェットは、ユーザが入力したり、表示するテキスト情報をエンコードするために、現在のロケールのコードセットを使用します。ウィジェットは、正しいフォントでデータを表示するために、次のいずれかの検索パターンを使用してフォントを見つけます。ウィジェットは優先度の高い順に検索パターンを使用し、フォントセットが決まったところで検索をやめます。

1. フォント・リスト・エレメント・タグ `XmFONTLIST_DEFAULT_TAG` を持つフォントセットのエントリを、フォント・リスト中で検索する。
2. フォントセットを指定するエントリをフォント・リスト中で検索し、最初に見つかったエントリを使用する。
3. フォント・リスト中の最初のフォントを使用する。

テキスト・ウィジェットを通じて利用できる国際化機能は、初期の OSF/Motif リリースと比較すると、次の 2 点が変更されています。

- コンパウンド・ストリングのセグメントに、複数の文字セットのデータを含めることができる。

この機能は、フォントセット構造の導入と、言語ごとに 1 つの文字セットをサポートするのではなく、ロケールに必要なコードセットをサポートすることにより可能になります。一般に、ラテン系コードセット以外のコードセットでは複数の文字セットがサポートされています。この機能を利用するには、アプリケーションは次の規則に従わなければなりません。

- コンパウンド・ストリングのセグメントを表示するためのリスト・エレメントとして、適切なフォントセットをフォント・リスト構造体で定義する。
  - 1 つのフォントではなく、適切なフォントセットに合ったタグをコンパウンド・ストリングに含める。
- アジア系言語の入力サーバを指定するときに、XmText および XmTextField ウィジェットで、On-the-Spot、Off-the-Spot、Over-the-Spot、および Root-Window 入力スタイル (前編集スタイル) も選択できる。

アプリケーション用のロケールごとのリソース・ファイルを作成するときに、XmNpreeditType リソースの優先度リストとして、入力スタイルを指定できます。

---

#### 注意

---

Off-the-Spot 前編集スタイルを選択した場合、入力ステータスと前編集のための領域 (通常はウィンドウの下部) を確保するために、アプリケーション・ウィンドウが拡大されます。このため、Off-the-Spot 入力スタイルを使用するすべてのアプリケーションでは、自動サイズ変更が有効になっていなければなりません。

アジアの国々で使用する X アプリケーションや Motif アプリケーションを作成し、入力ステータスおよび前編集の領域を頻繁

に使用する場合、ツールキットの機能を使用してアプリケーションの自動サイズ変更機能を無効にしないでください。

---

次の関数を使用して、複数の文字セットを含むコードセット用のコンパウンド・ストリングを作成できます。

- `XmStringCreate()`。テキストとフォント・リスト・エレメント・タグから構成されるコンパウンド・ストリングを作成します。
- `XmStringCreateLocalized()`。現在のロケールの文字コードでコンパウンド・ストリングを作成します。

---

#### 注意

---

ヘブライ語などの言語における、右から左方向への言語テキストの表示は、`DXmCSText` ウィジェットを通じてサポートされます。  
`XmText` ウィジェットと `XmTextField` ウィジェットは、左から右方向への表示だけをサポートします。

---

### 5.2.3 ウィジェット・クラスの国際化機能

次のウィジェット・クラスは、`XmText` ウィジェットと `XmTextField` ウィジェット (5.2.2 項を参照) を通じて、母国語の入力機能と表示機能をサポートします。

- `Command`
- `FileSelectionBox`
- `Label`
- `List`
- `MessageBox`
- `SelectionBox`

## 5.3 X ライブラリの国際化機能の使用

`libX11` ライブラリのリリース 5 以降、X コンソーシアムは、異なるロケールのデータを扱える X クライアントを開発するための新しい仕様を規定しま



した。新しい仕様は、さまざまな母国語でのデータ処理を可能にする標準 C ライブラリからなる、ANSI C のロケールモデルに基づいています。この仕様は、次のインタフェースを提供します。

- さまざまな母国語でのユーザ入力 of 要求
- 母国語のテキストに使用されるフォントの表示
- 言語固有のリソース値の取得
- コードセット変換を通じて母国語テキストをサポートするクライアント間通信

以降の項では、X ライブラリを使用して国際化プログラムを作成するさまざまな方法について説明します。

- ロケールの管理 ( 5.3.1 項)
- さまざまなロケールでのテキストの表示 ( 5.3.2 項)
- クライアント間通信の処理 ( 5.3.3 項)
- 地域化されたリソース・データベースの処理 ( 5.3.4 項)
- テキスト入力の処理 ( 5.3.5 項)

プログラミング手法 (特にテキスト入力に関連する手法) の例として、この章では、`ximdemo` というアプリケーションからの抜粋を示します。完全なソース・ファイルと、このアプリケーションの `Imakefile` が、ディレクトリ `$I18NPATH/usr/examples/ximdemo` にあります。このソース・ファイルを熟読して、アプリケーションを構築し実行することにより、この章で説明されているプログラミング手法の応用について理解をさらに深めることができます。

### 5.3.1 ロケールの管理

国際化対応の X クライアントは、他の種類のアプリケーションが使用するのと同じロケール宣言メカニズム、つまり、標準 C ライブラリの `setlocale` 関数を使用します。また、X ライブラリにはロケールを判定し、ロケール修飾子を構成するための 2 つの関数、`XSupportsLocale()` と `XSetLocaleModifiers()` が含まれています。表 5-1 に、これらの関数とその要約を示します。これらの関数の詳細については、`XSetLocaleModifiers(3X11)` のリファレンス・ページを参照してください。

表 5-1: X ライブラリのロケール宣言関数

関数	説明
XSupportsLocale()	X ライブラリが現在のロケールをサポートしているかどうかを判定します。
XSetLocaleModifiers()	現在のロケール設定での X 修飾子のリストを指定します。

XSetLocaleModifiers() リストは、次の形式の要素を持つ、ヌルで終了する文字列です。

@category =value

ロケール修飾子として現在定義されている標準 *category* は、入力サーバを識別するための *im* だけです。ただし、ロケールで複数の入力サーバがサポートされている場合は、複数の *im* エントリが修飾子リストに表示されることがあります。

ローカル・ホスト・システムでの省略値を設定するために、XMODIFIERS 環境変数に定義されている値が、XSetLocaleModifiers() 関数呼び出しで指定された修飾子のリストの後に追加されます。たとえば、Tru64 UNIX システムでは、入力サーバの省略値は DEC です。次のコマンドは、XMODIFIERS 変数にこの値を明示的に設定します。

% setenv XMODIFIERS @im=DEC

この例では、XSetLocaleModifiers() 関数の呼び出しで指定された修飾子リストの後に値 @im=DEC が追加されます。

X ライブラリ関数は、現在のロケールとロケール修飾子の設定値、あるいは、関数に渡されるオブジェクトにアタッチされているロケールとロケール修飾子の設定値に従って動作します。ロケール設定に関連するオブジェクトには、次のタイプがあります。

- テキスト入力に関連する XIM と XIC
- テキストの表示とサイズ調節に関連する XFontSet
- テキスト出力に関連する XOM と XOC

これらのオブジェクトは、アプリケーションのリソース・ファイルに関連する XrmDatabase の、バージョン 6 の実装で取り入れられました。

これらのオブジェクトのロケールとロケール修飾子は、オブジェクトが作成された時点のロケール設定に依存します。そのため、さまざまな言語の

オブジェクトを作成し、それらのオブジェクトを同時に使用することにより、複数のロケールのデータを処理できます。この機能により、多国語対応の X ウィンドウ・アプリケーションが開発できます。アプリケーションを開発するときは、次の規則に従ってください。

- データに適用されるロケールを識別し、そのデータを適切なロケール固有のオブジェクトで扱う。  
データのロケールと、オブジェクトのロケールが一致しない場合、結果は予測できません。
- 標準 C ライブラリの WPI インタフェース (たとえば、`printf()`) にテキストを渡す場合は、プロセスの現在のロケール設定と、渡すデータのロケールを一致させる。

例 5-1 に、X アプリケーションでロケールを設定したり、判定する方法を示します。

#### 例 5-1: X ウィンドウ・アプリケーションにおけるロケールの設定

```
#include <stdio.h>
#include <X11/Xlocale.h>
#include <X11/Xlib.h>
:
:

#define DEFAULT_LOCALE      "zh_TW.dechanyu"  1
:
:

main(argc, argv)
int    argc;
char   *argv[];
{
:
:
    immodifier[0]          = '\0';
    for(i=1; i<argc; i++) {
        if(!strcmp(argv[i], "-Root")) {
            best_style = XIMPreeditNothing;
        }
:
:
        else if (!strcmp(argv[i], "-locale"))  2
            locale = argv[++i];
        else if (!strcmp(argv[i], "-immodifier")) {
            strcpy(immodifier, "@im=");
            strcat(immodifier, argv[++i]);
        }
    }
:
:
    if(locale == NULL)
```

### 例 5-1: X ウィンドウ・アプリケーションにおけるロケールの設定 (続き)

```
        locale = DEFAULT_LOCALE; ❸
    if(setlocale(LC_CTYPE, locale) == NULL) {
        fprintf(stderr, "Error : setlocale() !\n");
        exit(0);
    }
    if (!XSupportsLocale()) {
        fprintf(stderr, "X does not support this locale");
        exit(1);
    }
    if (XSetLocaleModifiers(immodifier) == NULL) {
        (void) fprintf(stderr, "%s: Warning : cannot set locale \
modifiers. \n", argv[0]);
    }
    :
    :
```

- ❶ 省略時のロケールの設定値を格納する定数を定義します。

この例では、この定数の値は明示的に `zh_TW.dechanyu` に設定されています。

- ❷ アプリケーションのコマンド行でロケールが指定されているかどうかを判定します。

ユーザは、アプリケーションを起動するコマンド行で `-locale` オプションを指定することにより、省略時のロケールを無効にできます。

- ❸ アプリケーションのコマンド行でロケールが指定されていない場合、`DEFAULT_LOCALE` 定数の値をロケールに設定します。

この定数に `zh_TW.dechanyu` ではなく、ヌル文字列("") が設定されていた場合、省略時のロケールは、アプリケーションを実行しているプロセスの `LANG` 環境変数の設定値によって決まります。

### 5.3.2 さまざまなロケールでのテキストの表示

一部のロケールのコードセット、特にアジア系言語のコードセットでは、定義されているすべての文字を表示するためには、複数の X ウィンドウ・フォントが必要になります。そのようなコードセットを扱うために、X ライブラリは、テキストを表示したりそのサイズを調べるのに数種類のフォントを使用できる、フォントセットという概念をサポートしています。フォントセットの概念は、`XFontSet` 構造体によって実装されます。`XFontSet` 構造体

は、リリース 5 より前の X ライブラリでサポートされていた `XFontStruct` 構造体に置き換わるものです。

フォントセットは、フォントセットを作成したときのロケールにバインドされます。テキストの表示とサイズ調節を行う関数は、フォントセットのロケールに従ってテキストを解釈します。したがって、これらの関数は、文字をそのフォント・グリフに正しくマップします。

Tru64 UNIX の実装では、テキストの表示とサイズ調節を行う関数により、文字コードの異なる複数のフォントを使用して母国語テキストを表示できます。

5.3.2.1 フォントセットの作成と操作

表 5-2 に、フォントセットを作成したり操作する関数の要約を示します。詳細については、各関数のリファレンス・ページを参照してください。

表 5-2: フォントセットの作成と操作を行う X ライブラリ関数

関数	説明
<code>XCreateFontSet()</code>	指定されたディスプレイ用のフォントセットを作成します。この関数は、現在のロケールに必要なコードセットを判別し、それらのコードセットをサポートするための一連のフォントをロードします。
<code>XFreeFontSet()</code>	指定されたフォントセットと、ベース・フォント名リスト、フォント名リスト、 <code>XFontStruct</code> リスト、 <code>XFontSetExtents</code> などの関連コンポーネントを解放します。
<code>XFontsOfFontSet()</code>	指定されたフォントセットの <code>XFontStruct</code> 構造体とフォント名のリストを返します。
<code>XBaseFontNameListOfFontSet()</code>	フォントセットの作成時にクライアントが指定した、オリジナルのベース・フォント名リストを返します。
<code>XLocaleOfFontSet()</code>	指定されたフォントセットにバインドされているロケールの名前を返します。

例 5-2 に、フォントセットを作成したり使用する関数を示します。

## 例 5-2: X ウィンドウ・アプリケーションにおけるフォントセットの作成と使用

```
...
#define DEFAULT_FONT_NAME      "-*-SCREEN-*-*-R-Normal--*-*," 1
...

char                *base_font_name = NULL;

...

XFontSet            font_set;

...

char                **missing_list;
int                 missing_count;
char                *def_string;

...

if (base_font_name == NULL)
    base_font_name = DEFAULT_FONT_NAME; 2
font_set = XCreateFontSet(display, base_font_name, &missing_list,
                        &missing_count, &def_string);

...

/*
 * if there are charsets for which no fonts can be found,
 * print a warning message.
 */
if (missing_count > 0) {
    fprintf(stderr, "The following charsets are \
missing: \n");
    for (i=0; i<missing_count; i++)
        fprintf(stderr, "%s \n", missing_list[i]);
    XFreeStringList(missing_list);
}

...
```

1 DEFAULT\_FONT\_NAME 定数を，省略時のベース・フォント名リストの値として定義します。

この例では，省略時のベース・フォント名リストとして，  
-\*-SCREEN-\*-\*-R-Normal--\*-\*，-\* を設定します。省略時のベース・フォント名リストには，フォントを完全に指定した形式のリストではなく，総称名（この例のようにワイルドカード・フィールドを使用）を指定します。完全に指定した形式のフォント・リストは特定のロケールだけでしか使用できませんが，総称名は複数のロケールで省略値として使用できます。

- ② 省略時のベース・フォント名リストが、コマンド行で指定されているかどうかを判定します。

アプリケーションのコマンド行で `-fs` オプションを使用すると、省略時のベース・フォント名リストを無効にできます。

### 5.3.2.2 フォントセットのメトリックスの取得

表 5-3 に、フォントセットのメトリックスを調べたり、テキストのサイズを調節する X ライブラリ関数の要約を示します。

表 5-3: テキストのサイズを調節する X ライブラリ関数

関数	説明
<code>XExtentsOfFontSet()</code>	指定されたフォントセットのフォントの境界ボックス情報が格納されている <code>XFontSetExtents</code> 構造体を返します。
<code>XmbTextEscapement()</code> , <code>XwcTextEscapement()</code>	指定されたフォントセットを使用して、特定の文字列を表示するために必要な文字送り幅 (ピクセル数) を計算します。
<code>XmbTextExtents()</code> , <code>XwcTextExtents()</code>	文字列イメージ全体の境界ボックスと、スペーシングのための論理的な境界ボックスを計算します。これらの関数は、それぞれ <code>XmbTextEscapement()</code> と <code>XwcTextEscapement()</code> から返される値も返します。
<code>XmbTextPerCharExtents()</code> , <code>XwcTextPerCharExtents()</code>	指定されたフォントセット用にロードされたフォントに従って、指定されたテキスト中の各文字のサイズを返します。

### 5.3.2.3 フォントセットを使用したテキストの表示

表 5-4 に、さまざまな母国語でテキストを表示するための関数の要約を示します。テキストを表示する他の X ライブラリ関数とは異なり、国際化関数は次のような機能を備えています。

- 単一のフォントではなく、フォントセットを扱う。
- フォントセットのロケールに応じてテキストを表示する。

アプリケーションはこれらの関数を使用することで、テキストのエンコードを直接処理しないようにします。

表 5-4: テキストを表示するための X ライブラリ関数

関数	説明
<code>XmbDrawText()</code> , <code>XwcDrawText()</code>	複数のフォントセットを使用してテキストを表示します。複雑なスペーシングや、テキスト文字列間でのフォントセットの切り替えを可能にします。 これらの関数は、対応する単一フォント用の関数、 <code>XDrawText()</code> と <code>XDrawText16()</code> の代わりに使用します。
<code>XmbDrawString()</code> , <code>XwcDrawString()</code>	1 つのフォントセットを使用して、指定されたテキストだけをフォアグラウンド・ピクセルで表示します。 これらの関数は、対応する単一フォント用の関数、 <code>XDrawString</code> と <code>XDrawString16</code> の代わりに使用します。
<code>XmbDrawImageString()</code> , <code>XwcDrawImageString()</code>	表示する長方形をバックグラウンド・ピクセルで塗り潰します。その後、指定されたイメージ・テキストを 1 つのフォントセットを使用して表示し、そのテキストをフォアグラウンド・ピクセルでペイントします。 これらの関数は、対応する単一フォント用の関数、 <code>XDrawImageString()</code> と <code>XDrawImageString16()</code> の代わりに使用します。

例 5-3 に、国際化関数でテキストを表示する方法を示します。

例 5-3: X ウィンドウ・アプリケーションにおけるテキストの表示

<pre>GC      Jxgc_on, Jxgc_off; int      Jxcx, Jxcy; int      Jxcx_offset=2, Jxcy_offset=2; int      Jxsfont_w, Jxwfont_w, Jxfont_height; XRectangle *Jxfont_rect; int      Jxw_width, Jxw_height; #define Jxmax_line      10 int      Jxsize[Jxmax_line]; char      Jxbuff[Jxmax_line][128]; int      Jxline_no; int      Jxline_height; : : static int</pre>	
--	--



### 例 5-3: X ウィンドウ・アプリケーションにおけるテキストの表示 (続き)

```
JxWriteText(display, client, font_set, len, string)
{
    Display *display;
    Window   client;
    XFontSet font_set;
    int       len;
    char      *string;
    {
        int fy;
        XFillRectangle(display, client, Jxgc_off, Jxcx, Jxcy,
                        Jxsfont_w, Jxfont_height); 1
        if(len == 1 &&
            (string[0] == LF || string[0] == TAB
             || string[0] == CR)) {
            _JxNextLine();
            XFillRectangle(display, client, Jxgc_off, 0, Jxcy,
                            Jxw_width, Jxfont_height);
        }
        else {
            if(Jxcx >= (Jxw_width - Jxwfont_w)
               || (Jxsize[Jxline_no] + len) >= 256) {
                _JxNextLine();
                XFillRectangle(display, client, Jxgc_off, 0, Jxcy,
                                Jxw_width, Jxfont_height);
            }
            strncpy(&Jxbuff[Jxline_no][Jxsize[Jxline_no]], string,
                    len);
            Jxsize[Jxline_no] += len;
            fy = -Jxfont_rect->y + Jxcy;
            XmbDrawImageString(display, client, font_set,
                                Jxgc_on, Jxcx, fy, string, len); 2
            Jxcx += XmbTextEscapement(font_set, string, len); 3
            if(Jxcx >= Jxw_width) {
                _JxNextLine();
                XFillRectangle(display, client, Jxgc_off, 0, Jxcy, \
                                Jxw_width, Jxfont_height);
            }
        }
        XFillRectangle(display, client, Jxgc_on, Jxcx, Jxcy, \
                        Jxsfont_w, Jxfont_height);
    }
}
```

- 1 XFillRectangle() を使用して、ブロック・タイプのカーソルを表示します。
- 2 XmbDrawImageString() を使用して、母国語の文字列を表示します。

この文字列には、シングルバイト文字とマルチバイト文字の両方が含まれていることがあります。

③ `XmbTextEscapement()` を使用して、次の文字列の表示位置を計算します。

5.3.2.4 X 出力サーバを使用したテキスト処理

前項で説明したフォントセットの概念は、X ライブラリのバージョン 5 で取り入れられました。X ライブラリのバージョン 6 では、出力サーバと出力コンテキストをさらに汎用化した概念が実装されています。出力サーバと出力コンテキストは、フォントとコンテキストに依存した処理を実行し、左右両方向のテキスト表示やコンテキストに応じたテキスト出力を可能にします。

ロケールに従ってテキストを表示するためには、アプリケーションは、そのテキストを表示するのに必要なフォント、テキストをコンポーネントに分離する方法、および各コンポーネントに必要なフォントについてのデータが必要です。このため、X ライブラリのバージョン 6 ではこの要件に対処するために、次のオブジェクトが用意されました。

- X 出力モジュール (XOM)  
XOM は、アプリケーションが出力サーバと通信するのに使用できるデータ構造体です。
- X 出力コンテキスト (XOC)  
XOC は、プログラム・インタフェースの観点からは `XFontSet` と互換性がありますが、より汎用化されたオブジェクトです。

表 5-5 に、XOM と XOC に関連する X ライブラリ関数をまとめます。これらの X ライブラリ関数についての詳細は、それぞれのリファレンス・ページを参照してください。

表 5-5: 出力サーバと出力コンテキスト用の X ライブラリ関数

関数	説明
<code>XOpenOM()</code>	現在のロケールと修飾子の指定に合った出力サーバをオープンします。この関数は、現在のロケールと修飾子に対応する XOM オブジェクトを返します。
<code>XCloseOM()</code>	指定された出力サーバをクローズします。
<code>XSetOMValues()</code>	出力サーバの属性を設定します。

表 5-5: 出力サーバと出力コンテキスト用の X ライブラリ関数 (続き)

関数	説明
XGetOMValues()	指定された出力サーバのプロパティまたは機能を取得します。
XDisplayOfOM()	指定された出力サーバに対応するディスプレイを返します。
XLocaleOfOM()	指定された出力サーバに対応するロケールを返します。
XCreateOC()	指定された出力サーバ内に出力コンテキストを作成します。
XOMOfOC()	指定された出力コンテキストに対応する出力サーバを返します。
XSetOCValues()	XOC オブジェクトの値を設定します。
XGetOCValues()	XOC オブジェクトの値を取得します。
XDestroyOC()	指定された出力コンテキストを破棄します。

### 5.3.2.5 エンコーディングが異なるフォントセット間の変換

次の理由により、エンコーディングの異なる複数の X フォントが利用できることがあります。

- 1 つの文字セットで複数のエンコーディングが一般的に使用されるため  
たとえば、日本語 (JIS X0208)、中国語 (GB 2312)、および韓国語 (KS C 5601) の文字セットでは、GL エンコーディングと GR エンコーディングの両方が使用できます。
- 1 つの国で複数の文字セットがサポートされることがあるため
- 製品に採用されているフォント・エンコーディングがベンダごとに異なるため

システムごとにフォント・エンコーディングが異なると、種類の違うシステムで実行されるアプリケーションでは問題が発生します。このため、テキストの表示とサイズ調節用の関数の実装には、フォント・エンコーディングを変換するためのメカニズムが組み込まれています。変換を可能にするには、アプリケーションが実行時環境に適したベース・フォント名リストを決定できるようにしなければなりません。アプリケーションは、リソース・ファイルから、あるいは、コマンド行でユーザが指定したオプションにより、ベー

ス・フォント名リストを取得できます。たとえば、`ximdemo` アプリケーションを実行するコマンド行では、`-fs` オプションを使用して、ベース・フォント名リストを指定できます。

フォント・エンコーディングのための変換メカニズムは、アプリケーションが X ライブラリの国際化テキスト出力関数を使用している場合にのみ利用できます。この変換メカニズムは、`XDrawText()` や `XDrawString()` などの基本的なテキスト出力関数では使用できません。

5.3.3 クライアント間通信の処理

さまざまな言語や国で使われるアプリケーションを設計する場合は、Latin-1 または ASCII テキストの文字列だけがクライアント間通信に使用されるという前提は成り立ちません。このため、X ライブラリにはクライアント間通信で任意の言語のテキスト文字列を扱える関数が含まれています。表 5-6 に、それらの関数の要約を示します。

表 5-6: クライアント間通信用の X ライブラリ関数

関数	説明
<code>XmbSetWMProperties()</code>	ウィンドウの主要なプロパティを設定する唯一のプログラミング・インタフェースです。アプリケーションはこれらのプロパティを使用して、他のクライアント、特にウィンドウ・マネージャおよびセッション・マネージャと通信します。たとえば、この関数はウィンドウ名とアイコン名用の引数を持ちますが、一部のロケールではこの名前にマルチバイト文字を含めることができます。
<code>XmbTextListToTextProperty()</code> , <code>XwcTextListToTextProperty()</code>	現在のロケールでエンコードされているテキストを、 <code>STRING</code> 型または <code>COMPOUND_TEXT</code> 型のテキスト・プロパティに変換します。
<code>XmbTextPropertyToTextList()</code> , <code>XwcTextPropertyToTextList()</code>	<code>STRING</code> 型または <code>COMPOUND_TEXT</code> 型のテキスト・プロパティを、マルチバイト文字列またはワイド文字列のリストに変換します。

表 5-6: クライアント間通信用の X ライブラリ関数 (続き)

関数	説明
<code>XwcFreeStringList()</code>	<code>XwcTextPropertyToTextList()</code> によって割り当てられたメモリを解放します。
<code>XDefaultString()</code>	文字が変換できなかった場合の置き換えとして、省略時の文字列を調べます。 変換ルーチンは、変換できない文字を含む文字列を検出した場合、ロケールごとの省略時の文字列に置き換えます。 <code>XDefaultString()</code> 関数は、この省略時の文字列を調べます。

例 5-4 は、X アプリケーションにおけるクライアント間通信の例です。

#### 例 5-4: X ウィンドウ・アプリケーションにおける他のクライアントとの通信

```

:
    if (!strcmp(locale, "zh_TW.dechanyu")) {
        strcpy(title, "XIM F|n/");
    } else if (!strcmp(locale, "zh_CN.dechanzi")) {
        strcpy(title, "XIM J>76");
    } else if (!strncmp(locale, "ja_JP", 5)) {
        strcpy(title, "XIM %G%b");
    } else if (!strcmp(locale, "ko_KR.deckorean")) {
        strcpy(title, "XIM 5%8p");
    } else if (!strcmp(locale, "th_TH.TACTIS")) {
        strcpy(title, "XIM !RCJR8T5");
    } else {
        strcpy(title, "XIM Demo")  ❶
    }
    XmbSetWMPProperties(display, window, title, title, NULL, \
                        0, NULL, NULL, NULL);  ❷
:

```

- ❶ `strcmp()` 関数や `strcpy()` 関数の引数として、引用符で囲まれた母国語テキストを指定します。

この例では、テキストはウィンドウ・タイトルです。わかりやすくするために、関数呼び出しではテキスト文字列を明示的に指定しています。実際には、X や Motif のアプリケーションはこのような文字列を、ロ

ケール固有のリソース・ファイルまたはユーザ・インタフェース言語 (UIL) ファイルから取り出します。

- 2 XmbSetWMProperties() 関数にテキストを渡し、ロケールを使用してタイトルを解析して、その結果に応じてウィンドウ・マネージャのプロパティを設定します。

5.3.4 地域化されたリソース・データベースの処理

X リソース・ファイルのロケールは、ファイルが作成された時点のロケール設定に依存します。そのため、リソース・データベースを作成するためにロードされるリソース・ファイルや文字列は、現在のロケールで解析されます。この状況は、5.3.2 項のロケールとフォントセットのバインドで説明した状況と似ています。

表 5-7 に、地域化されたリソース・データベースを扱う X ライブラリ関数をまとめます。

表 5-7: 地域化されたリソース・データベースを扱う X ライブラリ関数

関数	説明
XrmLocaleOfDatabase()	指定されたデータベースにバインドされているロケール名を返します。
XrmGetFileDatabase()	指定されたファイルをオープンし、新しいリソース・データベースを作成して、ファイルから読み取った仕様をデータベースにロードします。 ファイルは、現在のロケールで解析されます。
XrmGetStringDatabase()	新しいリソース・データベースを作成し、ヌルで終了する文字列で指定されたリソースを格納します。 文字列は、現在のロケールで解析されます。
XrmPutLineResource()	指定されたデータベースに、リソース・エントリを 1 つ追加します。 エントリの文字列は、データベースのロケールで解析されます。

表 5-7: 地域化されたリソース・データベースを扱う X ライブラリ関数 (続き)

関数	説明
<code>XrmPutFileDatabase()</code>	指定されたデータベースのコピーを、指定されたファイルに格納します。 このファイルは、データベースと同じロケールで書き込まれます。
<code>XResourceManagerString()</code>	STRING 型でエンコードされた RESOURCE_MANAGER プロパティを、現在のロケールでエンコードされたマルチバイト文字列に変換します。 この関数は、 <code>XmbTextPropertyToTextList()</code> 関数と同じ方法でエンコーディングを変換します。

### 5.3.5 X 入力サーバを使用したテキスト入力の処理

国際化対応の X アプリケーションを開発する場合、1 つのキーボードから、さまざまなロケールでデータを入力できなければなりません。この問題に対処するために、次のオブジェクトが X ライブラリに取り入れられました。

- X 入力モジュール (XIM)

XIM は、アプリケーションが入力サーバと通信するのに使用できるデータ構造体です。

- X 入力コンテキスト (XIC)

XIC は、マルチスレッド対応のユーザ入力環境におけるテキスト入力フィールドの状態を表します。

アプリケーションは、ユーザがテキスト・データを入力できる複数のテキスト入力フィールドを用意することができます。フィールドは、切り換えることができます。入力データを取得するために、アプリケーションは入力コンテキストを指定して `XmbLookupString()` または `XwcLookupString()` を呼び出します。返される文字列は、常に XIM または XIC オブジェクトに対応するロケールでエンコードされています。以降の項では、入力オブジェクトの使用方法について説明します。

#### 5.3.5.1 入力サーバへの接続と、接続の解除

入力サーバを使用するためには、アプリケーションはまず、`XOpenIM()` を呼び出さなければなりません。この関数は、現在のロケールとロケール修飾子に対応した入力サーバへの接続を確立します。この関数は、現在のロケー

ルとロケール修飾子がバインドされている XIM オブジェクトを返します。  
ロケールと修飾子の XIM オブジェクトへのバインドは、この呼び出しを実行した時点で確立されるため、動的に変更することはできません。

入力サーバを必要としなくなった場合、アプリケーションは `XCloseIM()` を呼び出して XIM オブジェクトをクローズします。

XIM オブジェクトの情報を取得するためには、次の関数も利用できます。

- `XDisplayOfIM()`  
指定された XIM オブジェクトに対応するディスプレイを返します。
- `XLocaleOfIM()`  
指定された XIM オブジェクトに対応するロケールを返します。

`XOpenIM()` 関数によってオープンされる入力サーバは、次のいずれかの方法で決定されます (優先順位の高いものから順に示す)。

1. `XSetLocaleModifiers()` の呼び出しで指定した `im` 修飾子の値
2. `XMODIFIERS` 環境変数で指定した入力サーバ
3. `DEC` という名前の省略時の入力サーバ

上記の入力サーバが使用できない場合、`XOpenIM()` は省略時の動作として、ISO Latin-1 の入力だけをサポートします。次の場合には、`XOpenIM()` 呼び出しは失敗する可能性があります。

- 指定した入力サーバが動作していない場合
- 指定した `im` 修飾子が間違っている場合
- 指定した入力サーバが現在のロケールをサポートしていない場合

例 5-5 は、入力システムを接続する方法と接続の解除を行う方法の例です。

#### 例 5-5: X ウィンドウ・アプリケーションにおける入力サーバへの接続と接続の解除

---

```
main(argc, argv)
int      argc;
char     *argv[];
{
    Display *display;

    :

    XIM      im;
```



### 例 5-5: X ウィンドウ・アプリケーションにおける入力サーバへの接続と接続の解除 (続き)

```
⋮
⋮
char                *res_file = NULL;
⋮
⋮
XrmDatabase         rdb = NULL;
⋮
⋮
preedcb_cd.win = client;
if(res_file) {
    printf("Set Database : file name = %s\n", res_file);
    rdb = XrmGetFileDatabase(res_file); ❶
}
if((im = XOpenIM(display, rdb, NULL, NULL)) == NULL) {
    printf("Error : XOpenIM() !\n"); ❷
    exit(0);
}
⋮
⋮
XCloseIM(im); ❸
⋮
⋮
```

❶ 入力サーバに固有のリソースを探すために、リソース・データベース rdb を XOpenIM() に渡します。

国際化 Xt 関数により、アプリケーションで作成したリソース・データベースを指定できます。

❷ 入力サーバへの接続に成功したかどうかを検査します。

❸ 入力サーバとの接続を解除します。

#### 5.3.5.2 入力サーバ値の問い合わせ

入力サーバの動作には、ベンダによって異なる部分があります。たとえば実装によっては、サポートされるユーザ入力スタイルの組み合わせが異なることがあります。

移植性のあるアプリケーションの開発を支援するために、X ライブラリには入力サーバの属性を調べるための XGetIMValues() 関数が含まれています。XNQueryInputStyle 属性には、入力サーバがサポートしているユーザ入力スタイルを指定します。

例 5-6 に、XGetIMValues() 関数と XNQueryInputStyle 属性を使用して、入力サーバに関する情報を取得する方法を示します。

#### 例 5-6: 入力サーバがサポートするユーザ入力スタイルの取得

```
main(argc, argv)
int    argc;
char   *argv[];
{
    Display      *display;

    :

    int          i, n;

    :

    XIMStyles    *im_styles;
    XIMStyle      xim_mode=0;
    XIMStyle      best_style = XIMPreeditCallbacks;
    XIM           im;

    :

    XIMStyle      app_supported_styles;

    :

    for(i=1; i<argc; i++) {
        if(!strcmp(argv[i], "-Root")) {
            best_style = XIMPreeditNothing;
        }
        else if (!strcmp(argv[i], "-Cb")) {
            best_style = XIMPreeditCallbacks;  ❶
        }
    }

    :

    /* set flags for the styles our application can support */
    app_supported_styles = XIMPreeditNone | XIMPreeditNothing |
XIMPreeditCallbacks;  ❷
    app_supported_styles |= XIMStatusNone | XIMStatusNothing;
    XGetIMValues(im, XNQueryInputStyle, &im_styles, NULL);
    n = 1;  ❸
    if(im_styles != (XIMStyles *)NULL) {
        for(i=0; i<im_styles->count_styles; i++) {
            xim_mode = im_styles->supported_styles[i];
            if((xim_mode & app_supported_styles) ==
xim_mode) { /* if we can handle it */
                n = 0;
                if (xim_mode & best_style) /* pick user
selected style */
                    break;  ❹
            }
        }
    }
    if(n) {
        printf("warning : Unsupport InputStyle. or No
IMserver.\n");
        exit (0);
    }

    :
}
```

#### 例 5-6: 入力サーバがサポートするユーザ入力スタイルの取得 (続き)

---

- 1 ユーザがアプリケーションのコマンド行で、使用したい入力スタイルを指定したかどうかを判定します。

ximdemo アプリケーションでは、`-Root` と `-Cb` オプションを使用して、ユーザ入力スタイルを指定できます。これらのオプションは、このアプリケーションでサポートされている 2 つのスタイルを表します。  
`-Root` オプションは、`Root-window` スタイルを指定します。このスタイルでは、クライアントと入力サーバ間のやり取りを最小限に抑えることができます。`-Cb` オプションは、コールバックを介して前編集を処理するスタイルを指定します。このスタイルでは、`On-the-Spot` 前編集が可能になります。

- 2 アプリケーションがサポートできる 2 種類の入力スタイルを指定するために、`app_supported_styles` ビットマスクを定義します。

- 3 `XGetIMValues()` を呼び出して、入力スタイルを調べます。

この呼び出しは、入力スタイルを `im_styles` パラメータに返します。

- 4 入力サーバがサポートし、このアプリケーションが正しく処理できる入力スタイルを選択します。

ユーザが指定した入力スタイルが優先されます。入力スタイルが指定されていない場合、アプリケーションは、返されたスタイル・リスト中の最後の入力スタイルを選択します。

入力サーバでサポートされる入力スタイル (前編集スタイル) は、ロケールごとに異なります。

特定の入力サーバでサポートされている入力スタイルを見つけるには、次の各国のマニュアルを参照してください。

- 『*Technical Reference for Using Chinese Features*』
- 『日本語機能ガイドブック』
- 『*Technical Reference for Using Korean Features*』
- 『*Technical Reference for Using Thai Features*』

これらのマニュアルは、本オペレーティング・システムのドキュメント Web サイト (<http://tru64unix.compaq.co.jp/document/index.html>) で、プログラミング関連ドキュメントとして掲載されています。

### 5.3.5.3 入力サーバのコンテキストの作成と使用

X サーバが 1 つのディスプレイ上に複数のウィンドウを持てるのと同様に、アプリケーションも 1 つの入力サーバに対して複数のコンテキストを作成できます。X ライブラリには、入力コンテキスト (XIC) のオブジェクトを作成するための `XCreateIC()` 関数が含まれています。XIC オブジェクトには、他の関数を使用して設定したり取得できるいくつかの属性があります。そのような属性を次に示します。

- 入力コンテキストの入力スタイル
- 前編集とステータス・テキストの表示に使用されるフォントセット
- On-the-Spot 前編集を処理するためのコールバック

XIC オブジェクトを破棄するには、`XDestroyIC()` 関数を呼び出します。

例 5-7 に、`XCreateIC()` 関数と `XDestroyIC()` 関数の使用方法を示します。

#### 例 5-7: X ウィンドウ・アプリケーションにおける入力コンテキストの作成と破棄

---

```
⋮
    Display          *display;
⋮

    Window          root, window, client;
⋮

    XIMStyle         xim_mode=0;
⋮

    XIM              im;
    XIC              ic;
⋮

    XVaNestedList    preedit_attr, status_attr;
    XIMCallback      ximapicb[10];
    char             immodifier[100];
    preedcb_data     preedcb_cd;
⋮
⋮
```

### 例 5-7: X ウィンドウ・アプリケーションにおける入力コンテキストの作成と破棄 (続き)

```
    window = XCreateSimpleWindow(display, root, 0, 0,
                                W_WIDTH, W_HEIGHT, 2, bpixel, fpixel);
    :
    :

    client = JxCreateTextWindow(display, window, 0, 0,
                                W_WIDTH-2, W_HEIGHT-2, 1, bpixel, fpixel,
                                font_set, &font_height);
    :
    :

    if (xim_mode & XIMPreeditCallbacks) {
        ximapicb[0].client_data = (XPointer)NULL;
        ximapicb[0].callback = (XIMProc)api_preedit_start_cb;
        ximapicb[1].client_data = (XPointer)&preedit_cd;
        ximapicb[1].callback = (XIMProc)api_preedit_done_cb;
        ximapicb[2].client_data = (XPointer)&preedit_cd;
        ximapicb[2].callback = (XIMProc)api_preedit_draw_cb;
        ximapicb[3].client_data = (XPointer)NULL;
        ximapicb[3].callback = (XIMProc)api_preedit_caret_cb;
        nestlist = XVaCreateNestedList(10,
                                        XNPreeditStartCallback, &ximapicb[0],
                                        XNPreeditDoneCallback, &ximapicb[1],
                                        XNPreeditDrawCallback, &ximapicb[2],
                                        XNPreeditCaretCallback, &ximapicb[3],
                                        NULL); ①
    }
    if (xim_mode & XIMPreeditCallbacks) { ②
        ic = XCreateIC(im,
                      XNInputStyle, xim_mode,
                      XNClientWindow, window,
                      XNFocusWindow, client,
                      XNPreeditAttributes, nestlist,
                      NULL); ③
    } else {
        /* preedit nothing */
        ic = XCreateIC(im,
                      XNInputStyle, xim_mode,
                      XNClientWindow, window,
                      XNFocusWindow, client,
                      NULL ); ④
    }
    if(ic == NULL) { ⑤
        printf("Error : XCreateIC() !\n");
        XCloseIM(im);
        exit(0);
    }
    :
    :
exit:
    XDestroyIC(ic); ⑥
```

① `XVaCreateNestedList()` 関数を呼び出して、前編集属性とステータス属性の、ネストされた引数リストを作成します。

XNPreeditAttributes 属性と XNStatusAttributes 属性は、2 次属性のリストを含みます。2 次属性の設定や問い合わせを実行する前に、アプリケーションは、2 次属性を格納するためのネストされたリストを作成しなければなりません。

② XIC 属性を指定します。

XIC オブジェクトを作成するときに、アプリケーションはいくつかの XIC 属性を指定しなければなりません。XNInputStyle 属性は必須です。他の属性が必要かどうかは、入力スタイルによって異なります。

③ On-the-Spot 入力スタイルのコールバックを登録します。

入力スタイルが On-the-Spot の場合、アプリケーションは XIC オブジェクトの作成時に、すべてのコールバックを登録しなければなりません。

アプリケーションは、XIC の作成時に XNClientWindow 属性を設定する必要はありませんが、XIC を使用する前にこの属性を設定しなければなりません。XNClientWindow を設定する前に XIC を使用すると、結果は予測できません。

④ Root-Window スタイルの入力スタイル属性、クライアント・ウィンドウ属性、およびフォーカス・ウィンドウ属性を設定します。

入力スタイルが Root-Window の場合、XIC の作成時にアプリケーションが設定する必要のある属性は、これらの属性だけです。

⑤ XIC の作成に失敗したときの動作を指定します。

次の場合には、XCreateIC() 呼び出しは失敗します (つまり、NULL を返す)。

- 必須の属性が設定されていない場合
- 読み取り専用属性 (たとえば XNFilterEvents) が設定されている場合
- 属性名が認識できない場合

⑥ XIC をクローズします。

表 5-8 に、XIC オブジェクトの管理に利用できる関数の要約を示します。

表 5-8: 入力コンテキスト (XIC) を管理する X ライブラリ関数

関数	説明
XSetICFocus ( )	キーボード・イベントが入力サーバに送られるようにします。 XIC のフォーカス・ウィンドウが入力フォーカスを受け取ったときは、この関数を呼び出さなければなりません。この関数を呼び出さないと、キーボード・イベントは入力サーバに送られません。
XUnsetICFocus ( )	キーボード・イベントを入力サーバに送らないようにします。 XIC のフォーカス・ウィンドウがフォーカスを解除されたときに、この関数を呼び出します。
XmbResetIC ( ) , XwcResetIC ( )	XIC をリセットして初期状態に戻します。 XIC 上で保留されている入力はすべて削除されます。これらの関数は、現在の前編集文字列または NULL を返します。どちらが返されるかは、入力サーバの実装によって異なります。
XIMOfIC ( )	指定された XIC に対応する XIM を返します。
XSetICValues ( )	指定された XIC に属性を設定します。
XGetICValues ( )	指定された XIC の属性を調べます。

5.3.5.4 On-the-spot 入力スタイル用の前編集コールバックの作成

アプリケーションが On-the-Spot 入力スタイルをサポートしているときは、一連の前編集コールバックを用意しなければなりません。XIC には多数のコールバックが関連付けられています。例 5-8 に、それらのコールバックを示します。

例 5-8: X ウィンドウ・アプリケーションにおける前編集コールバックの使用

<pre>... int      Jxsize[Jxmax_line]; char     Jxbuff[Jxmax_line][128]; int      Jxline_no; int      Jxline_height; int      sav_cx, sav_cy; int      sav_w_width, w_height; int      sav_size[Jxmax_line]; int      sav_line_no; char     preedit_buffer[12]; void</pre>	
---	--

### 例 5-8: X ウィンドウ・アプリケーションにおける前編集コールバックの使用 (続き)

---

```
save_value()
{
    int i;
    sav_cx = Jxcx;
    sav_cy = Jxcy;
    sav_line_no = Jxline_no;
    for (i=0; i< Jxmax_line; i++)
        sav_size[i] = Jxsize[i];
}

void
restore_value()
{
    int i;
    Jxcx = sav_cx;
    Jxcy = sav_cy;
    Jxline_no = sav_line_no;
    for (i=0; i< Jxmax_line; i++)
        Jxsize[i] = sav_size[i];
}

int
api_preedit_start_cb(ic, clientdata, calldata)
XIC ic;
XPointer clientdata;
XPointer calldata;
{
    int len;
    len = 12;
    /* save up the values */
    save_value(); ①
    return(len); ②
}

void
api_preedit_done_cb(ic, clientdata, calldata)
XIC ic;
XPointer clientdata;
XPointer calldata;
{
    preedcb_data *cd = (preedcb_data *)clientdata;
    /* restore up the values */
    restore_value(); ③
    /* convenient handling */
    JxRedisplayText(cd->dpy, cd->win, cd->fset);
    return;
}

void
api_preedit_draw_cb( ic, clientdata, calldata)
XIC ic;
XPointer clientdata;
XIMPreeditDrawCallbackStruct *calldata;
{
    preedcb_data *cd = (preedcb_data *)clientdata;
    int count;
    char *reset_str;
    if (calldata->text) {
        if (calldata->text->encoding_is_wchar) ④
        {
        } else {
            count = strlen(calldata->text->string.multi_byte);
            if (count > 12) {

```



## 例 5-8: X ウィンドウ・アプリケーションにおける前編集コールバックの使用 (続き)

```
/* preedit string > max preedit buffer */
reset_str = XmbResetIC(ic); ⑤
XFillRectangle(cd->dpy, cd->win, Jxgc_off, Jxcx, Jxcy,
Jxw_width*13, Jxfont_height); /* clear the preedit area */
restore_value();
if (reset_str)
    XFree(reset_str);
return;
}
if (!calldata->chg_length) { /* insert character */
    if (!calldata->chg_first) { /* insert in first character
in preedit buffer */
strncpy(&preedit_buffer[0], calldata->text->string.multi_byte, count);
        restore_value();
    } else {
        /* Not Yet Implemented */
    }
} else { /* replace character */
    if (!calldata->chg_first) { /* replace from first
character in pre-edit buffer */
strncpy(&preedit_buffer[0], calldata->text->string.multi_byte, count);
        restore_value();
    } else {
        /* Not Yet Implemented */
    }
}
XFillRectangle(cd->dpy, cd->win, Jxgc_off, Jxcx, Jxcy,
Jxw_width*13, Jxfont_height); /* clear the preedit area */
JxWriteText(cd->dpy, cd->win, cd->fset, count, preedit_buffer);
} else { /* should delete preedit buffer */
    /* Not yet implemented */
}
return;
}
void
api_preedit_caret_cb(ic, clientdata, calldata)
XIC ic;
XPointer clientdata;
XIMPreeditCaretCallbackStruct *calldata;
{
    /* Not yet implemented */
    return;
}
:
```

### ① 現在の表示位置を保存します。

前編集文字列の表示操作の一環として、このアプリケーションは、現在の表示位置を `PreeditStartCallback` 属性の値として保存します。前

編集が完了すると、アプリケーションは前編集文字列を削除し、元の表示位置を復元します。

- ② 前編集文字列の長さを返します。

12 バイトという値は、文字列の長さを制限するためのものです。この値は、前編集バッファのサイズと同じでなければなりません。このアプリケーションでは、前編集バッファ (`preedit_buffer`) を 12 バイトの文字配列として宣言しています。

- ③ 表示位置を復元し、テキスト・バッファを再表示します。

- ④ ワイド文字のエンコーディングを処理します。

この例では、前編集文字列はマルチバイト・エンコーディングされているものとします。ただしアプリケーションは、マルチバイト・エンコーディングとワイド文字エンコーディングの両方を処理できるように作成しなければなりません。文字位置などの情報が、バイト数ではなく文字数として `XIMPreeditDrawCallbackStruct` 構造体に返されるため、ワイド文字エンコーディングの方が好ましいエンコーディングです。

- ⑤ 前編集文字列のサイズが 12 バイトを超えた場合は、前編集文字列をクリアします。

文字列のサイズは、`PreeditDrawCallback` 属性から取得します。アプリケーションは、`XmbResetIC()` 呼び出しで返された文字列を処理せずに、`Xfree()` を呼び出して解放します。

#### 5.3.5.5 入力サーバのためのイベント・フィルタリング

入力サーバは、アプリケーションによってイベントが処理される前に、イベントを受信しなければなりません。アプリケーションは、`KeyPress` イベントと `KeyRelease` イベントだけでなく、他のイベントも入力サーバに渡さなければなりません。X ライブラリには、イベントを入力サーバに渡すための `XfilterEvent()` 関数が含まれています。この関数は、他の関連する関数とともに次のように使用します。

1. `XNFilterEvents` 引数を指定して `XGetICValues()` 関数を呼び出し、入力サーバに渡すイベントのマスクを取得します。
2. `XSelectInput()` 関数によりイベント・タイプを登録します。
3. プログラムのメイン・ループ (通常は `XNextEvent()` 呼び出しの直後) で、`XfilterEvent()` を呼び出して入力サーバにイベントを渡します。

True のリターン・ステータスは、入力サーバがイベントをフィルタ処理したため、アプリケーションではそのイベントをこれ以上処理する必要がないことを示します。

例 5-9 に、上記の処理を示します。

#### 例 5-9: X ウィンドウ・アプリケーションにおける入力サーバのためのイベント・フィルタリング

---

```

:      long                im_event_mask;
:
:
:
:      XGetICValues(ic, XNFilterEvents, &im_event_mask, NULL);
:      mask = StructureNotifyMask | FocusChangeMask | ExposureMask;
:      XSelectInput(display, window, mask);
:      mask = ExposureMask | KeyPressMask | FocusChangeMask |
:             im_event_mask;
:      XSelectInput(display, client, mask);
:
:
:
:      for(;;) {
:          XNextEvent(display, &event);
:          if(XFilterEvent(&event, NULL) == True)
:              continue; ①
:          switch(event.type) {
:              /* dispatch event */
:
:
:
:          }
:      }
:
:
:

```

---

#### ① イベントをフィルタ処理します。

XtDispatchEvent() 関数は、XFilterEvent() を呼び出します。  
したがって、この例の for ループを、XtAppMainLoop() の呼び出しに置き換えることも可能です。

#### 5.3.5.6 キーボードからのコンパウンド・ストリングの取得

X アプリケーションで XmbLookupString() 関数や XwcLookupString() 関数を使用すると、母国語の文字とキー・シンボルを取得できます。アプリケーションでは、1 文字を組み立てるのにキーを数回押す必要がある、複雑な入力システムが存在することも考慮しておかなければなりません。そのた

め、これらの関数を呼び出すたびに、コンパウンド・キャラクタやコンパウンド・ストリングが返されることを前提にはできません。

例 5-10 に、X アプリケーションでキーボード入力を取得する方法を示します。

#### 例 5-10: X ウィンドウ・アプリケーションにおけるキーボード入力の取得

```
:
:
XEvent          event;
:
:
:
int              len = 128;
char             string[128];
KeySym          keysym;
int             count;
:
:
for(;;) {
    XNextEvent(display, &event);
    if(XFilterEvent(&event, NULL) == True)
        continue;
    switch(event.type) {
    case FocusIn : [1]
        if(event.xany.window == window)
            XSetInputFocus(display, client,
                RevertToParent, CurrentTime);
        else if(event.xany.window == client) {
            XSetICFocus(ic);
        }
        break;
    case FocusOut : [1]
        if(event.xany.window == client) {
            XUnsetICFocus(ic);
        }
        break;
    case Expose :
        if(event.xany.window == client)
            JxRedisplayText(display, client,
                font_set);
        break;
    case KeyPress : [2]
        count = XmbLookupString(ic, (XKeyPressedEvent
            *)&event, string, len, &keysym, NULL);
        if( count == 1 && string[0] == (0x1F&'c')) {
            /* exit */
            goto exit;
        }
        if( count > 0 ) { [3]
            JxWriteText(display, client,
                font_set, count, string);
        }
        break;
    case MappingNotify :
        XRefreshKeyboardMapping( (XMappingEvent *)&event);
        break;
    case DestroyNotify :
        printf("Error : DestroyEvent !\n");
        break;
    }
```

例 5-10: X ウィンドウ・アプリケーションにおけるキーボード入力の取得 (続き)

---

```
    }  
}
```

---

① FocusIn イベントと FocusOut イベントを処理します。

この例では、1 つの XIC がフォーカス・ウィンドウに対応しています。入力サーバの中には、ステータス領域を更新するためにフォーカス変更情報を必要とするものがあります。そのため、FocusIn イベントのたびに XSetICFocus() を呼び出し、また FocusOut イベントのたびに XUnsetICFocus() を呼び出します。

また、アプリケーションは、1 つの XIC を複数のフォーカス・ウィンドウに使用することもできます。この場合、フォーカス変更イベントごとに XSetICFocus() を呼び出す必要はありませんが、XIC に XNFocusWindow 属性を設定しなければなりません。

② KeyPress イベントを処理します。

アプリケーションでは、XmbLookupString() または XwcLookupString() に対して、KeyPress イベントだけを渡すようにします。これらの関数に KeyRelease イベントを渡した場合、結果は予測できません。

この例ではわかりやすくするために、XmbLookupString() 呼び出しのステータス・フィールドを NULL にしています。実際のアプリケーションでは、返されたステータスを検査して、適切に対処しなければなりません。たとえば、返されたステータスが XBufferOverflow であれば、アプリケーションではバッファに割り当てるメモリを追加します。

③ 文字列が返された場合は、その文字列を処理します。

XmbLookupString() は、コンパウンド・ストリングのサイズ (バイト数) を返します。

### 5.3.5.7 入力サーバが失敗したときの処理

入力サーバの XNDestroyCallback リソースと入力コンテキストは、X11R6 で取り入れられました。入力サーバが失敗したときに生成されるこれらのリソースは、クライアント・アプリケーションの XIM および XIC オブジェ

クトをクローズします。クライアント・アプリケーションがサーバの失敗を検出せずに動作を続行し、XIC および XIM オブジェクトをクローズした場合、結果は予測できません。

例 5-11 に、XIM オブジェクトの XNDestroyCallback リソースの登録方法と、サーバが失敗したときに XIM をクローズする方法を示します。

#### 例 5-11: 入力サーバが失敗したときの処理

```
static void      _imDestroyCallback(); [1]
:
:

      Bool                      IMS_Connected = False;
      XIMCallback              cb; [2]
:
:

      if((im = XOpenIM(display, rdb, NULL, NULL)) != NULL) {
          printf("Error : XOpenIM() !\n");
          exit(0);
      }
      else {
          IMS_Connected = True;
          cb.client_data = (XPointer) &IMS_Connected;
          cb.callback = (XIMProc) _imDestroyCallback;
          XSetIMValues(im, XNDestroyCallback, &cb, NULL); [3]
      }
:
:

      case KeyPress :
          if (IMS_Connected) count = XmbLookupString(ic,
(XKeyPressedEvent *)&event, string, len, &keysym, NULL);
          else count =
XLookupString((XKeyPressedEvent *)&event, string, len, &keysym, NULL); [4]
:
:

static void
_imDestroyCallback(im, client_data, call_data)
    XIM im;
    XPointer client_data;
    XPointer call_data;
{
    Bool *Connected = (Bool *)client_data;
    *Connected = 3D False; [5]
}
```

- [1] 何らかの原因で入力サーバ (IMS) が失敗した場合に XIM をクローズする関数を宣言します。
- [2] 入力サーバがまだ接続されているかどうかを示す IMS\_Connected 変数と、リソースの登録に必要なクライアント情報が格納される cb 構造体を宣言します。

- ③ XIM をオープンする呼び出しに失敗した場合は、エラー・メッセージを表示して終了します。

成功した場合には、IMS\_Connected 変数に True を設定し、適切なクライアント・データを cb 構造体に格納した後、XSetIMValues() 関数を呼び出して XIM の XNDestroyCallback リソースを登録します。

- ④ 入力サーバが動作している場合、XmbLookupString() 関数を呼び出して、ユーザ入力进行处理します。動作していない場合には、XLookupString() 関数を呼び出します。
- ⑤ 入力サーバが失敗した場合に XIM をクローズする関数のプロトタイプを定義します。

ximdemo プログラムは極めて簡単なものなので、入力コンテキストは 1 つしか使用しません。このような場合、入力サーバが失敗したときに、XIC を明示的にクローズする必要はありません。次の例は、XIC をクローズするコールバック関数のプロトタイプを示します。

```
static void icDestroyCallback(ic, client_data, call_data)
XIC ic;
XPointer client_data;
XPointer call_data;
```

### 5.3.6 Xt および X ライブラリの国際化機能の使用法: 要約

以下に示す母国語入力処理の手順は、X ライブラリについてこれまでの項で説明した内容の要約です。わかりやすくするために、手順の説明では X ツールキット・イントリンシクス・ライブラリ (Xt) 関数を使用するプログラミングと、X ライブラリ関数を使用するプログラミングの間に相違点があれば、その旨を明記します。X ツールキット・イントリンシクス・ライブラリの国際化機能の説明については、5.1 節を参照してください。

1. setlocale() を呼び出して、現在のロケールにバインドします。  
XtSetLanguageProc() で初期化コールバック関数を登録しても、同じ結果が得られます。
2. XSupportsLocale() を呼び出して、X が現在のロケールをサポートしていることを確認します。
3. XSetLocaleModifiers() を呼び出すか XMODIFIERS 環境変数を設定して、使用する入力サーバを定義します。
4. XOpenIM() を呼び出して、選択した入力サーバに接続します。

ウィジェットを作成する場合は、適切な XIM がリソースとしてウィジェットに渡されるため、この手順を省略できます。

5. `XGetIMValues()` を呼び出して、入力サーバがサポートしている入力スタイルを調べます。

ウィジェットを作成する場合、この処理は初期化関数内で行います。

6. XIC に対応するウィンドウを作成します。

Xt 関数を使用している場合は、ウィジェットを作成します。

7. `XCreateFontSet()` を呼び出して、このウィンドウのフォントセットを作成します。X11R6 では、代わりに `XOpenOM()` を使用できます。

Xt 関数を使用しており、ウィジェットをすでに作成してある場合は、`XtDefaultFontSet` の値を使用します。

8. アプリケーションで取得した、サポートされている入力スタイルの値から入力スタイルを選択し、その値を引数として `XCreateIC()` を呼び出します。

XIMPreeditCallbacks を使用している場合は、コールバック・ルーチンを作成して、`XCreateIC()` の呼び出しによりそれらのルーチンを登録しなければなりません。

9. `XGetICValues()` を呼び出して、`XNFilterEvents` 属性を調べ、入力サーバが必要とするフォーカス・ウィンドウからのイベントを登録します。

10. イベントをディスパッチする前に、メインのイベント・ループで `XFilterEvent()` を呼び出します。

この呼び出しで `True` が返された場合は、そのイベントを破棄できます。

X イントリンシクス (Xt) ライブラリのルーチンを使用したプログラミングでは、`XtDispatchEvent()` を使用します。

11. メインのイベント・ループで、フォーカス・ウィンドウが `FocusIn` イベントを受信したときは入力フォーカスを設定し、`FocusOut` イベントを受信したときは入力フォーカスの設定を解除します。

X イントリンシクス (Xt) ライブラリのルーチンを使用したプログラミングでは、イベント・ハンドラまたはトランスレーション/アクション・テーブルを使用して、フォーカス・イベントを処理します。



12. フィルタ処理されていない KeyPress イベントに対しては，  
XmbLookupString() または XwcLookupString() を呼び出して，  
キー・シンボルとコンパウンド・ストリングを取得します。  
このストリングは，テキスト表示用の国際化関数を使用して表示できます。



## ロケールの作成

この章では、ロケールの作成方法について説明します。ロケールは、母国語、文化的データ、オペレーティング・システム上のコードセットの特定の組み合わせをサポートするデータのセットです。localedef コマンドを使用して、以下のファイルからロケールを作成できます。

- 文字マップ・ソース・ファイル charmap (6.1 節)

このファイルのフォーマットや規則については、charmap(4) のリファレンス・ページを参照してください。この章では、すべての文字を 8 ビットのシングルバイトとして規定している、ISO Latin-1 コードセットのバイナリ文字コードに準拠する charmap ファイルの例を示します。またこの章では、シングルバイト文字とマルチバイト文字の両方を規定している SJIS コードセットの charmap ファイルの一部も示します。

- ロケール・ソース・ファイル (6.2 節)

このファイルのフォーマットや規則については、locale(4) のリファレンス・ページを参照してください。この章の例では、フランスの言語や習慣をサポートする fr\_FR.ISO8859-1@example というロケールを作成します。

- メソッド・ファイルおよび関連するシェアード・ライブラリ (6.3 節)

メソッド・ファイルとシェアード・ライブラリは、charmap ファイルでマルチバイト文字を定義する際に必要です。それ以外の場合、これらのファイルは省略可能です。メソッド・ファイルには、ロケールが使用し、関連するシェアード・ライブラリ内で定義されている各関数のエントリが入っています。メッセージ・ファイルのエントリには、ライブラリ名とパスが入っています。メソッド・ファイルのエントリは、データ・コードと内部処理 (ワイド文字) コード間の変換を行う C ライブラリ・インタフェースの再定義を含むシェアード・ライブラリも指定します。

デスクトップ・アプリケーションで新しいロケールを使用するために変更しなければならないファイルの一覧は、第 5 章を参照してください。

## 6.1 ロケールに対応した文字マップ・ソース・ファイルの作成

charmap ファイルは、文字のバイナリ・コードにシンボルを定義します。localedef コマンドは charmap ファイルを使用して、ロケール・ソース・ファイル内の文字シンボルを文字コードにマップします。例 6-1 は、この章で作成する fr\_FR.ISO8859-1@example ロケール用のソース・ファイル、ISO8859-1.cmap の一部です。D.1 節には、このファイルの内容がすべて示されています。

例 6-1: サンプル・ロケール用の charmap ファイル

```
# [1]
# Charmap for ISO 8859-1 codeset [1]
# [1]

<code_set_name>          "ISO8859-1" [2]
<mb_cur_max>             1 [2]
<mb_cur_min>             1 [2]
<escape_char>            \ [2]
<comment_char>           # [2]

CHARMAP [3]

# Portable characters and other standard [1]
# control characters [1]

<NUL>                    \x00 [4]
<SOH>                    \x01
<STX>                    \x02
<ETX>                    \x03
<EOT>                    \x04
<ENQ>                    \x05
<ACK>                    \x06
<BEL>                    \x07
<alert>                  \x07
<backspace>              \x08
<tab>                    \x09
<newline>                \x0a
<vertical-tab>           \x0b
<form-feed>              \x0c
<carriage-return>        \x0d
<SO>                     \x0e
:
:

<zero>                   \x30 [4]
<one>                    \x31
<two>                    \x32
```

### 6-2 ロケールの作成

## 例 6-1: サンプル・ロケール用の charmap ファイル (続き)

---

```
<three>                                \x33
<A>                                     \x41
<B>                                     \x42
<C>                                     \x43
<D>                                     \x44
:
:

<underscore>                           \x5f  4
<low-line>                              \x5f
<grave-accent>                          \x60
<a>                                       \x61
<b>                                       \x62
<c>                                       \x63
<d>                                       \x64
:
:

# Extended control characters            1
# (names taken from ISO 6429)           1

<PAD>                                    \x80  4
<HOP>                                    \x81
<BPH>                                    \x82
<NBH>                                    \x83
<IND>                                    \x84
:
:

# Other graphic characters              1

<nobreakspace>                          \xa0  4
<inverted-exclamation-mark>            \xa1
:
:

END CHARMAP                             5
```

---

### 1 コメント行

省略時のコメント文字は番号記号 (#) です。省略時の指定は、  
<comment\_char> 定義 (例 6-1 を参照) を使用して変更できます。

## ② キーワード宣言

この例では、有効な宣言をすべて示し、`<code_set_name>` を除くすべてのエントリに省略時の値を指定します。一般に、省略時の値を変更したい場合にのみ、宣言を指定します。この例では、`<comment_char>` と `<escape_char>` 宣言により、コメント文字とエスケープ文字の省略時の値を指定しています。`<mb_cur_max>` の値、つまりバイト単位での文字の最大の長さは、この charmap ファイルでは 1 です。`<mb_cur_min>` の値、つまりバイト単位での文字の最小の長さは、すべてのロケールの charmap ファイルで 1 でなければなりません (すべてのロケールには、シングルバイト文字を定義するポータブル文字セットの文字が含まれています)。

`<code_set_name>` の値は、実行時にロケールにバインドされるアプリケーションから呼び出される `nl_langinfo(CODESET)` によって返される値になります。

## ③ 文字マップの開始を示すヘッダ

## ④ 文字のシンボルからコードへのマップ

個々の文字マップは、シンボリック名とコード値から構成されます。名前とコード値は、1 つ以上のスペースで区切られます。

シンボリック名は、左山カッコ (<) で始まり、右山カッコ (>) で終わります。山カッコで囲まれる文字には、制御文字とスペース文字を除く、ポータブル文字セット中の任意の文字を使用できます。名前に複数の右山カッコ (>) が含まれている場合、最後の山カッコを除くすべての山カッコの前に `<escape_character>` の値を付けなければなりません。シンボリック名の長さは、128 バイトを超えることはできません。

コード値は、1 つまたは複数の 10 進数、8 進数、あるいは 16 進数の定数です (複数の定数はマルチバイトのコード値に対応します)。定数のフォーマットは次のとおりです。

- 10 進数

`\dnnn` または `\dnn`。 `n` は 10 進数です。

- 16 進数

`\xnn`。 `n` は 16 進数です。

- 8 進数

`\nnn` または `\nn`。 `n` は 8 進数です。

同じ文字 (コード値) に対して (各々, 異なるシンボリック名で) 複数の文字マップ・エントリを定義できます。この例では, 同じコード値に対して複数のシンボリック名は定義していません。

#### ⑤ 文字マップの終わりを示すトレイラ

マルチバイト文字を含むコードセット用のソース・ファイルは, より複雑な文字マップを含みます。例 6-2 は, 日本語の SJIS コードセット用のソース・ファイルに含まれる文字マップ・エントリの一部です。このソース・ファイルには, 同じコードセット中でサポートしなければならない複数の文字セットのエントリが指定されています。

#### 例 6-2: マルチバイト・コードセット用の **charmap** ファイルの一部

```
# SJIS charmap
#
<code_set_name> "SJIS" ①
<mb_cur_min>      1 ②
<mb_cur_max>      2 ③
CHARMAP
#
# CS0: ASCII
#
:

<commercial-at>      \x40 ④
<A>                   \x41 ④
<B>                   \x42 ④
:

#
# CS1:  JIS X0208-1983 for ShiftJIS.
#
<zenkaku-space>      \x81\x40 ⑤
<j0101>...<j0163>    \x81\x40 ⑤
<j0164>...<j0194>    \x81\x80 ⑤
:

#
# UDC Area in JIS X0208 plane
#
<u8501>...<u8563>    \xeb\x40 ⑥
<u8564>...<u8594>    \xeb\x80 ⑥
<u8601>...<u8663>    \xeb\x9f ⑥
:
```

## 例 6-2: マルチバイト・コードセット用の charmap ファイルの一部 (続き)

---

```
#
# CS2: JIS X0201 (so-called Hankaku-Kana)
#
<kana-fullstop>          \xa1  7
:
:
<kana-conjunctive>       \xa5  7
<kana-WO>                 \xa6  7
<kana-a>                  \xa7  7
:
:
END CHARMAP
```

---

### ① コードセット名

### ② 1 文字の最小バイト数

この値は 1 でなければなりません。

### ③ 1 文字の最大バイト数

SJIS では、マルチバイト文字の最大長は 2 バイトです。

### ④ ASCII 文字のシンボルとコード値

### ⑤ SJIS 文字のシンボルとコード値

文字シンボルを範囲として指定している点と、2 つの 16 進数値で 2 バイト文字のコード値を指定している点に注意してください。

シンボルがシンボル値の範囲として指定された場合、指定された文字コード値は、その範囲内の最初のシンボルに適用されます。localedef コマンドは、シンボル値とコード値の両方を自動的にインクリメントして、その範囲内のすべての文字のシンボル値とコード値を作成します。

### ⑥ SJIS コードセット中の UDC 用のマップ

これらのマップは、ユーザが後で文字を定義できるようにするための、コード値の範囲を設定します。

### ⑦ 半角カナ文字セット用のシングルバイト文字のマップ

文字マップ・ソース・ファイルに適用される規則の完全なリストについては、charmap(4) のリファレンス・ページを参照してください。



## 注意

文字マップ・ソース・ファイル内の文字のシンボリック名に関しては、標準化作業が進行中です。X/Open UNIX 標準の将来のバージョンでは、文字に関して長いシンボリック名と短いシンボリック名の両方が規定される予定です。

この例の文字のシンボリック名は、必ずしも各種の標準化団体が提唱している名前と一致しているわけではありません。

## 6.2 ロケール定義ソース・ファイルの作成

ロケール定義ソース・ファイルは、特定の言語と地域に固有のデータを定義します。ソース・ファイルはセクションに分けられており、定義するロケール・データのカテゴリごとに1つのセクションがあります。次のロケール・カテゴリがあります。

- LC\_CTYPE。文字クラスと属性を定義します (6.2.1 項)。
- LC\_COLLATE。文字や文字列の照合方法を定義します (6.2.2 項)。
- LC\_MESSAGES。肯定応答や否定応答に使用される文字列を定義します (6.2.3 項)。
- LC\_MONETARY。金額値の規則と記号を定義します (6.2.4 項)。
- LC\_NUMERIC。数値データの規則と記号を定義します (6.2.5 項)。
- LC\_TIME。日付と時刻を定義します (6.2.6 項)。
- LC\_ALL。すべてのカテゴリを指します。

例 6-3 に、ロケール定義ソース・ファイルの構成を擬似コードで示します。

例 6-3: ロケール・ソース定義ファイルの構成

```
# comment-line      1

comment_char        <char_symbol1>  2
escape_char         <char_symbol2>  3

CATEGORY_NAME      4

category_definition-statement  5
category_definition-statement  5
```

### 例 6-3: ロケール・ソース定義ファイルの構成 (続き)

---

```
⋮  
END CATEGORY_NAME 6  
⋮  
7
```

---

#### 1 コメント行

番号記号 (#) は、省略時のコメント文字です。行の最初のカラムにコメント文字を置くことにより、その行全体をコメントとして指定できます。ロケール・ソース・ファイル内では、定義文と同じ行にコメントを置くことはできません。ロケール・ソース・ファイルは、この点で文字マップ・ソース・ファイルと異なります。

#### 2 コメント文字の再定義

省略時のコメント文字は、`comment_char` キーワードで始まり、その後に使用したいコメント文字のシンボルが続くエントリ行で置き換えることができます。文字シンボルは、ロケールの文字マップ (`charmap`) ソース・ファイルで定義されています。

#### 3 エスケープ文字の再定義

省略時のエスケープ文字は、バックスラッシュ (\) です。この文字は、10 進、16 進、および 8 進の定数で使用され、定義文がソース・ファイルの次の行に続くことを示します。省略時のエスケープ文字は、`escape_char` キーワードで始まり、その後に 1 つまたは複数の空白文字、さらにその後に使用したいエスケープ文字のシンボルが続くエントリ行で置き換えることができます。文字シンボルは、そのロケールの文字マップソース・ファイルで定義されています。

#### 4 ロケール・カテゴリ・セクションのヘッダ

セクション・ヘッダは、`LC_CTYPE`、`LC_COLLATE`、`LC_NUMERIC`、`LC_MONETARY`、`LC_MESSAGES`、および `LC_TIME` の各カテゴリ名に対応しています。

## 5 カテゴリの定義文

これらの文のフォーマットは、カテゴリによって異なります。一般に、定義文はキーワードで始まり、その後に1つまたは複数のスペースあるいはタブが続き、最後に定義そのものが続きます。

カテゴリ定義文の代りに `copy` 文を記述して、他のロケール・ソース・ファイル内の定義文を取り込むことができます。たとえば、次のように記述します。

```
copy en_US.ISO8859-1
```

`copy` 文を記述するときには、カテゴリ内には他の文を記述しないでください。

## 6 ロケール・カテゴリ・セクションのトレイラ

セクション・トレイラは `END` キーワードで始まり、その後にカテゴリ名が続きます。

## 7 すべてのロケール・カテゴリのセクションを含めることも、一部のカテゴリのセクションだけを含めることもできます。ソース・ファイルでロケール・カテゴリのセクションを省略した場合、省略したカテゴリの定義は、省略時のロケール (POSIX または C ロケール) と同じになります。

以下の項では、特定のロケール・カテゴリに着目し、`fr_FR.ISO8859-1@example.src` ロケール・ソース・ファイルの個々の部分について説明します。D.2 節に、このソース・ファイル全体を示しています。

### 6.2.1 LC\_CTYPE ロケール・カテゴリの定義

ロケール・ソース・ファイルの `LC_CTYPE` セクションは、文字クラスと、大文字と小文字の変換などの操作に使用される文字属性を定義します。例 6-4 に、`LC_CTYPE` セクションの定義を示します。

#### 例 6-4: LC\_CTYPE カテゴリの定義

```
#####
LC_CTYPE 1
#####

upper  <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
        <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>;\
        <A-grave>;\

:

        <U-diaeresis> 2
```

## 例 6-4: LC\_CTYPE カテゴリの定義 (続き)

```
lower    <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
         <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>;\
         <a-grave>;\
:
:
         <u-diaeresis> 2

space    <tab>;<newline>;<vertical-tab>;<form-feed>;\
         <carriage-return>;<space> 2

cntrl    <NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;\
         <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
         <form-feed>;<carriage-return>;\
:
:
         <SOS>;<SGCI>;<SCI>;<CSI>;<ST>;<OSC>;<PM>;<APC> 2

graph    <exclamation-mark>;<quotation-mark>;<number-sign>;\
:
:
         <u-circumflex>;<u-diaeresis>;<y-acute>;<thorn-icelandic>;<y-diaeresis> 2

# print class includes everything in the graph class above, plus <space>.

print    <exclamation-mark>;<quotation-mark>;<number-sign>;\
:
:
         <u-circumflex>;<u-diaeresis>;<y-acute>;<thorn-icelandic>;<y-diaeresis>;\
         <space> 2

punct    <exclamation-mark>;<quotation-mark>;<number-sign>;\
         <dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;\
         <left-parenthesis>;<right-parenthesis>;<asterisk>;\
         <plus-sign>;<comma>;<hyphen>;<period>;<slash>;\
         <colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
         <greater-than-sign>;<question-mark>;<commercial-at>;\
         <left-square-bracket>;<backslash>;<right-square-bracket>;\
         <circumflex>;<underscore>;<grave-accent>;<left-brace>;\
         <vertical-line>;<right-brace>;<tilde> 2

digit    <zero>;<one>;<two>;<three>;<four>;\
         <five>;<six>;<seven>;<eight>;<nine> 2

xdigit   <zero>;<one>;<two>;<three>;<four>;\
         <five>;<six>;<seven>;<eight>;<nine>;\
         <A>;<B>;<C>;<D>;<E>;<F>;\
         <a>;<b>;<c>;<d>;<e>;<f> 2

blank    <space>;<tab> 2

toupper  (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
         (<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
         (<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
         (<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
         (<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
         (<z>,<Z>);\
```

## 例 6-4: LC\_CTYPE カテゴリの定義 (続き)

```
(<a-grave>,<A-grave>);\
(<a-circumflex>,<A-circumflex>);\
(<ae-ligature>,<AE-ligature>);\
(<c-cedilla>,<C-cedilla>);\
(<e-grave>,<E-grave>);\
(<e-acute>,<E-acute>);\
(<e-circumflex>,<E-circumflex>);\
(<e-diaeresis>,<E-diaeresis>);\
(<i-circumflex>,<I-circumflex>);\
(<i-diaeresis>,<I-diaeresis>);\
(<o-circumflex>,<O-circumflex>);\
(<u-grave>,<U-grave>);\
(<u-circumflex>,<U-circumflex>);\
(<u-diaeresis>,<U-diaeresis>) 3

# tolower class is the inverse of toupper.

tolower (<A>,<a>);(<B>,<b>);(<C>,<c>);(<D>,<d>);(<E>,<e>);\
(<F>,<f>);(<G>,<g>);(<H>,<h>);(<I>,<i>);(<J>,<j>);\
(<K>,<k>);(<L>,<l>);(<M>,<m>);(<N>,<n>);(<O>,<o>);\
(<P>,<p>);(<Q>,<q>);(<R>,<r>);(<S>,<s>);(<T>,<t>);\
(<U>,<u>);(<V>,<v>);(<W>,<w>);(<X>,<x>);(<Y>,<y>);\
(<Z>,<z>);\
(<A-grave>,<a-grave>);\
(<A-circumflex>,<a-circumflex>);\
(<AE-ligature>,<ae-ligature>);\
(<C-cedilla>,<c-cedilla>);\
(<E-grave>,<e-grave>);\
(<E-acute>,<e-acute>);\
(<E-circumflex>,<e-circumflex>);\
(<E-diaeresis>,<e-diaeresis>);\
(<I-circumflex>,<i-circumflex>);\
(<I-diaeresis>,<i-diaeresis>);\
(<O-circumflex>,<o-circumflex>);\
(<U-grave>,<u-grave>);\
(<U-circumflex>,<u-circumflex>);\
(<U-diaeresis>,<u-diaeresis>) 3

END LC_CTYPE 4
```

### 1 セクション・ヘッダ

### 2 文字クラスの定義

これらの定義は、文字クラスを表すキーワード (プロパティとも呼ぶ) で始まり、その後に 1 つまたは複数の空白文字と、そのクラスに含まれるすべての文字シMBOLのリストが続きます。文字シMBOLではなくコード値を指定することもできますが、その場合には、ロケール・ソース・ファイルの可読性が損われ、複数のコードセットで使用できなくなります。

例には示していませんが、水平方向の省略記号 (...) を使用して文字の範囲を表すことができます。たとえば文字列 `<NUL>;...;<tab>` では、省略記号 (...) は、コード値がシンボル `<NUL>` からシンボル `<tab>` までの範囲にあるすべての文字を表します。シンボルとそのコード値は、そのロケールの `charmap` ファイルで定義されています。

X/Open UNIX 標準で定義された文字クラスは、次のキーワードで表します。

- `upper` (大文字の英字)
- `lower` (小文字の英字)
- `alpha` (すべての英字)

省略時の定義では、`alpha` クラスは、`upper` クラスと `lower` クラスの文字を合わせたものです。サンプルのロケールでは `alpha` クラスを明示的に定義していないので、省略時の定義が適用されます。

- `space` (ホワイトスペース文字)
- `cntrl` (制御文字)
- `punct` (句読点)
- `digit` (数字)
- `xdigit` (16 進の数字)
- `blank` (空白文字)
- `graph` (スペース文字を除く印字可能文字)

省略時の定義では、このクラスは `alpha` , `digit` , および `punct` クラスの文字を合わせたものです。

- `print` (スペース文字を含む印字可能文字)

省略時の定義では、このクラスは `alpha` , `digit` , および `punct` クラスの文字と空白文字を合わせたものです。

アプリケーションの観点からは、`alnum` というクラスも存在します。このクラスは、`alpha` クラスと `digit` クラスの文字を合わせたものと定義されているため、ロケールで定義されることは稀です。

Unicode (\*.UTF-8) ロケールは、Unicode 標準で定義されている文字クラスを含みます。Unicode の文字分類についての詳細は、`locale(4)` を参照してください。

日本語などのアジア系言語のロケールでは、標準以外の文字クラスも定義されています。

### ③ 英字の大文字/小文字変換の定義

大文字/小文字変換の定義は、キーワード `toupper` と `tolower` で始まり、シンボルを個別に指定するのではなく対として指定します。この例の `toupper` の定義では、対の最初のシンボルが英字の小文字を、2 番目のシンボルが英字の大文字を表します。この定義は、関数 `towupper()` と `towlower()` がテキスト・データの大文字/小文字変換を実行するときの、変換対象の英字を定義します。

標準以外の文字クラスを定義するロケールでは、`wctrans()` と `towctrans()` 関数が使用する、その他のプロパティ変換も定義される場合があります。

### ④ セクション・トレイラ

上記の例は、`LC_CTYPE` カテゴリを定義するときに使用できるすべてのオプションを示しているわけではありません。その他のオプションを使用して、次の作業を行うこともできます。

- `copy` 文を使用して、他のロケールのカテゴリ定義全体をインクルードする。

`copy` 文を使用するときは、`copy` 文がセクション・ヘッダとトレイラ間の唯一のエントリでなければなりません。

- 標準文字クラスのいずれかを省略したり、別の文字クラスを定義する。

標準の文字分類は言語によって異なります。そのため、標準の文字クラスが適用できない言語もあります。ロケールを定義する場合は、そのロケールの言語に適した標準の文字クラスだけを使用してください。言語によっては、標準以外のクラス定義が必要になる場合があります。

標準以外の文字クラスを定義するには、そのクラスの前に `charclass` 文でクラスのキーワードを定義して、その後にクラス定義を続けます。次に例を示します。

```
charclass vowel
vowel      <a>;<e>;<i>;<o>;<u>;<y>
```

アプリケーションでは `wctype()` と `iswctype()` 関数を使用して、すべての文字クラス (ユーザ定義の文字クラスも含む) を判別し、テストできま

す。また、標準文字クラスのテストには、クラス固有の関数 `iswalpha()`、`iswpunct()` などを使用できます。

注意

`fr_FR.ISO8859-1@example` ロケールの `LC_CTYPE` カテゴリは、フランス語の文字に限定されています。一部のロケール開発者は、ISO 8859-1 文字セットでサポートされているすべての言語の文字を含めた文字クラスを定義しようとします。これによって、`copy` 文を使うことで、複数の西ヨーロッパ言語で同じ `LC_CTYPE` ソース定義を使用することができます。

`LC_CTYPE` カテゴリの定義に適用されるその他の規則と制約については、`locale(4)` のリファレンス・ページを参照してください。

6.2.2 LC\_COLLATE ロケール・カテゴリの定義

ロケール・ソース・ファイルの `LC_COLLATE` セクションには、文字と文字列の照合方法を指定します。例 6-5 は、`LC_COLLATE` セクションの一部です。

例 6-5: LC\_COLLATE カテゴリの定義

```
LC_COLLATE      1
order_start      forward;backward;forward  2
<NUL>           3
<SOH>
<STX>
<ETX>
<EOT>
<ENQ>
<ACK>
<alert>
<backspace>
<tab>
:
:

<APC>           3
<space>          <space>;<space>;<space>
<exclamation-mark> <exclamation-mark>;<exclamation-mark>;<exclamation-mark>
<quotation-mark>  <quotation-mark>;<quotation-mark>;<quotation-mark>
:
:

<a>              3
<A>              <a>;<a>;<a>
<feminine>       <a>;<a>;<A>
<a-acute>         <a>;<feminine>;<feminine>
<A-acute>         <a>;<a-acute>;<a-acute>
<a-grave>         <a>;<a-acute>;<A-acute>
<A-grave>         <a>;<a-grave>;<a-grave>
<A-grave>        <a>;<a-grave>;<A-grave>
```



## 例 6-5: LC\_COLLATE カテゴリの定義 (続き)

```
<a-circumflex>      <a>;<a-circumflex>;<a-circumflex>
<A-circumflex>      <a>;<a-circumflex>;<A-circumflex>
<a-ring>             <a>;<a-ring>;<a-ring>
<A-ring>             <a>;<a-ring>;<A-ring>
<a-diaeresis>        <a>;<a-diaeresis>;<a-diaeresis>
<A-diaeresis>        <a>;<a-diaeresis>;<A-diaeresis>
<a-tilde>            <a>;<a-tilde>;<a-tilde>
<A-tilde>            <a>;<a-tilde>;<A-tilde>
<ae-ligature>        <a>;<a><e>;<a><e>
<AE-ligature>        <a>;<a><e>;<A><E>
<b>                  <b>;<b>;<b>
<B>                  <b>;<b>;<B>
<c>                  <c>;<c>;<c>
<C>                  <c>;<c>;<C>
<c-cedilla>          <c>;<c-cedilla>;<c-cedilla>
<C-cedilla>          <c>;<c-cedilla>;<C-cedilla>
:
:
<z>                  <z>;<z>;<z>
<Z>                  <z>;<z>;<Z>
UNDEFINED
order_end
END LC_COLLATE
```

3

4

5

6

### 1 セクション・ヘッダ

### 2 要素に照合重みを割り当てる文が記述されているセクションの開始を示す order\_start キーワード

order\_start キーワードの行には、各ソート・パスに適用される、セミコロン (;) で区切られたソート・ディレクティブが続きます。ソート・ディレクティブには次のキーワードを指定できます。

- forward, 文字列の先頭から末尾に向かって比較操作を実行します。
- backward, 文字列の末尾から先頭に向かって比較操作を実行します。
- position, 照合重み IGNORE が適用されない文字列中の文字の相対位置を考慮して、比較操作を実行します。つまり、照合重みが IGNORE でなく、文字列の先頭 (forward, position) または末尾 (backward, position) から最も短い距離にある文字が最初に照合されます。

ソート・ディレクティブに 2 つのキーワードを指定する場合、つまり、`position` キーワードを `forward` または `backward` のいずれかと組み合わせて指定する場合、2 つのキーワードはコンマ (,) で区切ります。`position` キーワード自体は、ディレクティブ `forward, position` と同じ効果を持ちます。

ソート・ディレクティブの数は、それ以降の文で各照合要素に割り当てられる重みの数に対応します。

各ソート・ディレクティブと関連する重みのセットは、文字列比較の 1 回のパスあるいはレベルの情報を指定します。1 番目のディレクティブは、文字列比較操作で 1 次重みが適用されるときに使用され、2 番目のディレクティブは操作で 2 次重みが適用されるときに使用されます (3 番目以降のディレクティブについても同様)。文字列を正しく照合するのに必要なレベル数は、言語と文化習慣に依存するため、ロケールごとに異なります。また、レベル数には、`limits.h` と `sys/localedef.h` ファイルの `COLL_WEIGHTS_MAX` の設定によって決まる最大値があります。Tru64 UNIX システムでは、最大照合レベル (ソート・ディレクティブ) は 6 に制限されています。

`backward` ディレクティブは多くの言語で使用されるディレクティブであり、比較対象の文字列が等価な場合に、アクセントの付いた文字をアクセントの付かない文字の後に置くために使用します。

`position` ディレクティブは、西ヨーロッパ言語のハイフン (-) のように、単語内での位置によって意味が変わる文字を扱うために頻繁に使用されます。このディレクティブの使用例としては、“o-ring” という単語を単語リストの中で “or-ing” の前に並べたいが、ハイフンを考慮するのは、すべての文字列を英字だけでソートした後にしたい場合が挙げられます。このような順序にソートするためには、2 つのソート・ディレクティブと、それに関連する重み指定子のセットが必要です。最初の比較操作では、ソート・ディレクティブとして `forward` を指定し、すべての英字について英字を最初の重みとして使用して、ハイフン文字の重みとしては `IGNORE` を指定します。2 回目、あるいはそれ以降の比較操作では、ソート・ディレクティブとして `forward position` を指定し、すべての英字について `IGNORE` を重みとして使用して、ハイフン文字の重みとしてハイフンを指定します。

ソート・ディレクティブを指定しない場合、省略時のディレクティブには `forward` が使用されます。

### ③ 要素のための照合順序文

これらの文は、文字シンボルの後に、1 つまたは複数のブランク文字 (スペースまたはタブ) を指定し、さらにその後にソートの各段階で同じ重みを持つ文字のシンボルを指定します。

この例では、ソート順序は、制御文字、区切り文字と数字、英字の順です。英字は複数のパスでソートされ、最初のパスでは判別記号と大文字/小文字の区別は無視され、2 番目のパスで判別記号が有効になり、3 番目のパスで大文字/小文字の区別が有効になります。

### ④ 未定義文字のための照合順序文

UNDEFINED キーワードは、ロケールの charmap ファイルでは定義されているが、他の照合順序文では指定されていないすべての文字に適用される照合順序文を開始します。UNDEFINED カテゴリに対応する文字は、正規表現では、同じ等価クラスに属するものと見なされます。

UNDEFINED 照合順序文は必ず指定しておきます。指定しない場合には、localedef コマンドは未定義文字を照合順序の最後に含め、警告を出します。

また、UNDEFINED 文を最後の照合順序文として指定すると、localedef コマンドは、すべての未定義文字を 1 つのエントリに圧縮できる場合があります。これにより、ロケールのサイズを小さくできます。

このロケールは、ロケールの charmap ファイルに指定された任意の (ただし、他の照合順序文で扱わない) 文字の照合順序を最後に位置づけるように指定します。

UNDEFINED 文は、オペランドを持つことができます。たとえば、IGNORE キーワードを指定すると、他の照合順序文に指定されていない文字は、IGNORE が指定されているソート・パスで無視されます。この例に次のような UNDEFINED 文があったとすると、この文で指定されているソート・パスでは、他の照合順序文に指定されていない文字は無視されます。

```
UNDEFINED      IGNORE; IGNORE; IGNORE
```

### ⑤ 照合順序文の終わりを示すトレイラ

### ⑥ LC\_COLLATE セクションの終わりを示すトレイラ



```

collating-symbol <LOWERCASE>
collating-symbol <UNACCENTED>
:

order_start forward;backward;forward;forward
:

<UNACCENTED>
:

<LOWERCASE>
<a>          <a>;<UNACCENTED>;<LOWERCASE>;IGNORE
:

```

Unicode ロケールと dense コード・ロケールは等価なため、Unicode ロケールと dense コード・ロケールの両方で同じ charmap とロケール・ソースを使用することができます。ただし、charmap に定義されていて、LC\_COLLATE セクションに定義されていない Unicode 文字と dense コード文字は、異なる方法でソートされることがあります。

LC\_COLLATE カテゴリの定義についての詳細は、locale(4) のリファレンス・ページを参照してください。

### 6.2.3 LC\_MESSAGES ロケール・カテゴリの定義

ロケール・ソース・ファイルの LC\_MESSAGES セクションには、ユーザからの肯定応答と否定応答に使用する文字列を定義します。例 6-6 は、LC\_MESSAGES セクションです。

#### 例 6-6: LC\_MESSAGES カテゴリの定義

---

```

LC_MESSAGES      1

# yes expression. The following designates:
# "^[oO] [oO] [uU] [iI])"

yesexpr          "<circumflex><left-parenthesis>\
<left-square-bracket><o><O><right-square-bracket>\
<vertical-line><left-square-bracket><o><O>\
<right-square-bracket><left-square-bracket><u><U>\
<right-square-bracket><left-square-bracket><i><I>\
<right-square-bracket><right-parenthesis>"      2

# no expression. The following designates:

```

## 例 6-6: LC\_MESSAGES カテゴリの定義 (続き)

```
# "^( [nN] | [nN] [oO] [nN] ) "  
  
noexpr      "<circumflex><left-parenthesis>\<left-square-bracket><n><N><right-square-bracket>\<vertical-line><left-square-bracket><n><N>\<right-square-bracket><left-square-bracket><o><O>\<right-square-bracket><left-square-bracket><n><N>\<right-square-bracket><right-parenthesis>" 3  
  
# yes string. The following designates: "oui:o:O"  
  
yesstr      "<o><u><i><colon><o><colon><O>" 4  
  
# no string. The following designates: "non:n:N"  
  
nostr       "<n><o><n><colon><n><colon><N>" 5  
END LC_MESSAGES 6
```

### 1 セクション・ヘッダ

### 2 肯定応答として有効な表現の定義

このエントリは、`yesexpr` キーワードと、その後続く 1 つまたは複数のスペースあるいはタブ、さらにその後続く二重引用符で区切られた拡張正規表現で構成します。

この表現は、このロケールでは“oui”や“o”(大文字/小文字は無視される)が有効な肯定応答であることを指定しています。`yesexpr` の正規表現は、ロケールの `charmap` ファイルで定義されているシンボルを用いて、個々の文字を指定しています。

### 3 否定応答として有効な表現の定義

このエントリは、`noexpr` キーワードと、その後続く 1 つまたは複数のスペースあるいはタブ、さらにその後続く二重引用符で区切られた拡張正規表現で構成します。

この表現は、このロケールでは“non”や“n”(大文字/小文字は無視される)が有効な否定応答であることを指定しています。

### 4 肯定応答として有効な文字列の定義

このエントリは、`yesstr` キーワードと、その後続く 1 つまたは複数のスペースあるいはタブ、さらにその後続く二重引用符で区切られた文字列で構成します。

X/Open の UNIX 標準では、`yesstr` エントリは廃止される予定ですが、一部のアプリケーションやシステム・ソフトウェアではまだ、`yesexpr` ではなく `yesstr` が使用されている可能性があります。ロケールで `yesstr` を定義しておくことにより、そのようなソフトウェアでもロケールが使用できるようにしておくことが好ましい手法です。X/Open UNIX 標準では `yesstr` に対して固定文字列を 1 つだけ定義しています。複数の固定文字列を区切る区切り文字のコロン (:) は、標準仕様に対する拡張です。

#### ⑤ 否定応答として有効な文字列の定義

このエントリは、`nostr` キーワードと、その後続く 1 つまたは複数のスペースあるいはタブ、さらにその後続く二重引用符で区切られた文字列で構成します。

X/Open の UNIX 標準では、`nostr` エントリは廃止される予定ですが、一部のアプリケーションやシステム・ソフトウェアではまだ、`noexpr` ではなく `nostr` が使用されている可能性があります。ロケールで `nostr` を定義しておくことにより、そのようなソフトウェアでもロケールが使用できるようにしておくことが好ましい手法です。X/Open UNIX 標準では `nostr` に対して固定文字列を 1 つだけ定義しています。複数の固定文字列を区切る区切り文字のコロン (:) は、標準仕様に対する拡張です。

#### ⑥ セクション・トレイラ

シンボル定義を指定する代わりに、セクション・ヘッダとトレイラ間で `copy` 文を使用して、既存のロケールの `LC_MESSAGES` カテゴリの定義をコピーできます。`copy` 文はカテゴリの完全な定義であり、明示的なシンボル定義とともに使用することはできません。

### 6.2.4 LC\_MONETARY ロケール・カテゴリの定義

ロケール・ソース・ファイルの `LC_MONETARY` セクションには、金額値の書式に使用する規則とシンボルを定義します。アプリケーション開発者は `localeconv()` と `nl_langinfo()` 関数を使用して、このセクションで定義されている情報を取得し、`strfmon()` 関数により書式規則を適用します。例 6-7 は、`LC_MONETARY` セクションです。

## 例 6-7: LC\_MONETARY カテゴリの定義

```
LC_MONETARY 1

int_curr_symbol    "<F><R><F><space>" 2
currency_symbol    "<F>" 2
mon_decimal_point  "<comma>" 2
mon_thousands_sep  "" 2
mon_grouping       3;0 2
positive_sign      "" 2
negative_sign      "<hyphen>" 2
:
END LC_MONETARY 3
```

### 1 セクション・ヘッダ

### 2 シンボル定義

上記の例のエントリは、次の指定を行います。

- 国際通貨記号は FRF (フランスのフラン) とし、国内通貨記号は F (フラン) とする。
- コンマ (,) を小数点文字とする。
- ピリオド (.) を小数点左側の千単位の区切り文字とする。
- このロケールでは、小数点の左側の桁は 3 桁ごとにグループ化される。このロケールでは省略時の 1000 単位の通貨区切りを定義していないため、このロケールで定義されている通貨のグループ区切りは、アプリケーションが 1000 単位の区切りを指定する関数を使用したときのみ有効になります。
- 正符号は付けない。
- 負符号はマイナス (-) とする。

### 3 セクション・トレイラ

LC\_MONETARY セクションで定義可能なシンボル名を、以下に示します。

- int\_curr\_symbol  
国際通貨記号
- currency\_symbol



## 国内通貨記号

- `mon_decimal_point`  
金額値の書式で使用する小数点文字
- `mon_thousands_sep`  
小数点左側の桁を区切るための文字
- `mon_grouping`  
小数点の左側で桁をグループ化する際の桁数。 `mon_thousands_sep` で定義された文字がある場合は、 `mon_grouping` で定義されるグループの間にこの文字が挿入されます。 セミicolon (;) で区切って桁数を複数指定することで、グループの大きさを変化させることができます。たとえば、 `3;2` と指定すると、小数点の左側の最初のグループは 3 桁になり、それに続くグループはすべて 2 桁になります。 Tru64 UNIX システムでは、 `3;0` と `3` は同じであり、小数点の左側のすべての桁は、3 桁ごとにグループ化されます。
- `positive_sign`  
金額が負でないことを表す文字列
- `negative_sign`  
金額が負であることを表す文字列
- `int_frac_digits`  
書式中に `int_curr_symbol` があるときに、小数点右側に置かれる桁数
- `frac_digits`  
書式中に `currency_symbol` があるときに、小数点右側に置かれる桁数
- `p_cs_precedes`  
国際通貨記号または国内通貨記号を、負でない金額の前に置くかどうかを決定する整数
- `p_sep_by_space`  
国際通貨記号または国内通貨記号を、負の金額書式の他の部分からスペース文字で区切るかどうかを決定する整数
- `n_cs_precedes`

国際通貨記号または国内通貨記号を，負の金額書式の前に置くかどうかを決定する整数

- `n_sep_by_space`

国際通貨記号または国内通貨記号を，負の金額書式の他の部分からスペース文字で区切るかどうかを決定する整数

- `p_sign_posn`

負でない金額書式に正符号の文字列があるかどうかを判定し，ある場合にはその位置を示す整数

- `n_sign_posn`

負符号の文字列が負の金額書式中に現れる位置を示す整数

シンボル定義を指定する代わりに，セクション・ヘッダとトレイラの間で `copy` 文を使用して，既存のロケールの `LC_MONETARY` 定義をコピーできます。 `copy` 文はカテゴリの完全な定義であり，明示的なシンボル定義とともに使用することはできません。

`LC_MONETARY` の定義には，ユーロを完全にサポートしている UTF-8 および ISO8859-15 ロケールの言語のユーロ文字が設定されています。ユーロ文字は Latin-1 に入っていないため，ISO8859-1 ロケールの言語は，ユーロを採用していても，ユーロより前の通貨記号を使用します。たとえば，イタリア語のロケール `it_IT.ISO8859-15` は，ユーロをサポートします。イタリア語のロケール `it_IT.ISO8859-1` は，リラをサポートします。

`LC_MONETARY` シンボル定義についての詳細は，`locale(4)` のリファレンス・ページを参照してください。

### 6.2.5 `LC_NUMERIC` ロケール・カテゴリの定義

ロケール・ソース・ファイルの `LC_NUMERIC` セクションには，数値データを書式付けるための規則とシンボルを定義します。 `localeconv()` と `nl_langinfo()` 関数を使用することにより，この書式情報にアクセスできます。例 6-8 は，`LC_NUMERIC` セクションです。

### 例 6-8: LC\_NUMERIC カテゴリの定義

```
LC_NUMERIC      ❶  
decimal_point    "<comma>"  ❷  
thousands_sep   " "        ❸  
grouping         3;0        ❹  
  
END LC_NUMERIC  ❺
```

#### ❶ カテゴリ・ヘッダ

#### ❷ 小数点文字の定義

❸ 小数点の左側の桁を区切るために使用する文字の定義。このロケールでは、省略時の文字は定義されていません。したがって、アプリケーションでは必要に応じてこの文字を指定する必要があります。

❹ 小数点の左側の桁の各グループの大きさ。thousands\_sep で定義される文字があるときには、grouping で定義されるグループの間に挿入されます。

セミコロン (;) で区切って桁数を複数指定することで、グループの大きさを変化させることができます。たとえば、3;2 と指定すると、小数点の左側の最初のグループは3桁になり、それに続くグループはすべて2桁になります。Tru64 UNIX システムでは、3;0 と 3 は同じであり、小数点の左側のすべての桁は、3桁ごとにグループ化されます。

#### ❺ カテゴリ・トレイラ

例 6-8 は、LC\_NUMERIC セクションで定義可能なすべてのシンボルを示します。シンボル定義を使用する代わりに、セクション・ヘッダとトレイラ間に copy 文を指定して、他のロケールからこのセクションをインクルードできます。

LC\_NUMERIC シンボル定義の詳細については、locale(4) のリファレンス・ページを参照してください。

## 6.2.6 LC\_TIME ロケール・カテゴリの定義

ロケール・ソース・ファイルの LC\_TIME セクションには、date コマンドで使用されるフィールド記述子の解釈を定義します。このカテゴリ・セクションは、strftime(), wcsftime(), strptime(), および

nl\_langinfo() 関数の動作に影響します。例 6-9 に、サンプルのフランス語ロケールについて定義されているシンボルの一部を示します。

#### 例 6-9: LC\_TIME カテゴリの定義

```
LC_TIME 1

abday    "<d><i><m>";\
         "<l><u><n>";\
         "<m><a><r>";\
         "<m><e><r>";\
         "<j><e><u>";\
         "<v><e><n>";\
         "<s><a><m>" 2

day       "<d><i><m><a><n><c><h><e>";\
         "<l><u><n><d><i>";\
         "<m><a><r><d><i>";\
         "<m><e><r><c><r><e><d><i>";\
         "<j><e><u><d><i>";\
         "<v><e><n><d><r><e><d><i>";\
         "<s><a><m><e><d><i>" 3

abmon     "<j><a><n>";\
         "<f><e><acute><v>";\
         "<m><a><r>";\
         "<a><v><r>";\
         "<m><a><i>";\
         "<j><u><n>";\
         "<j><u><l>";\
         "<a><o><u><circumflex>";\
         "<s><e><p>";\
         "<o><c><t>";\
         "<n><o><v>";\
         "<d><e><acute><c>" 4

mon       "<j><a><n><v><i><e><r>";\
         "<f><e><acute><v><r><i><e><r>";\
         "<m><a><r><s>";\
         "<a><v><r><i><l>";\
         "<m><a><i>";\
         "<j><u><i><n>";\
         "<j><u><i><l><l><e><t>";\
         "<a><o><u><circumflex><t>";\
         "<s><e><p><t><e><m><b><r><e>";\
         "<o><c><t><o><b><r><e>";\
         "<n><o><v><e><m><b><r><e>";\
         "<d><e><acute><c><e><m><b><r><e>" 5
```

## 例 6-9: LC\_TIME カテゴリの定義 (続き)

```
# date/time format. The following designates this
# format: "%a %e %b %H:%M:%S %Z %Y"

d_t_fmt "<percent-sign><a><space><percent-sign><e>\
<space><percent-sign><b><space><percent-sign><H>\
<colon><percent-sign><M><colon><percent-sign><S>\
<space><percent-sign><Z><space><percent-sign><Y>" 6
:
:
END LC_TIME 7
```

### 1 セクション・ヘッダ

### 2 曜日の省略名

書式中にこの文字列を含めるには、%a 変換指定子を使用します。

### 3 曜日の完全名

書式中にこの文字列を含めるには、%A 変換指定子を使用します。

### 4 月の省略名

書式中にこの文字列を含めるには、%b 変換指定子を使用します。

### 5 月の完全名

書式中にこの文字列を含めるには、%B 変換指定子を使用します。

### 6 日付と時刻情報を組み合わせた書式

この書式では、strftime() 関数で定義されているように、フィールド記述子を結合します。フィールド記述子の一覧については、strftime(3) のリファレンス・ページを参照してください。

規定されている書式には、曜日の省略形 (%a)、月内の日付 (%e)、24 時間制での時間 (%H)、分 (%M)、秒 (%S)、タイム・ゾーン (%Z)、年の完全表現 (%Y) のフィールド記述子があります。米国東部時間で日付が 1999 年 4 月 23 日の場合、この例に指定された書式では、date コマンドで `ven 23 avr 13:43:05 EDT 1999` と表示されます。

### 7 セクション・トレイラ

例 6-9 は、LC\_TIME カテゴリのすべての標準シンボル定義を含んでいるわけではありません。LC\_TIME では、次の標準の定義を指定することもできます。

- d\_fmt  
日付だけの書式。 %x フィールド記述子に対応します。
- t\_fmt  
時刻だけの書式。 %X フィールド記述子に対応します。
- am\_pm  
午前と午後を表す文字列の書式。 %p フィールド記述子に対応します。  
たとえば、英語における定義は次のとおりです。  

```
am_pm          "<A><M>" ; "<P><M>"
```
- t\_fmt\_ampm  
12 時間制での時刻の書式。 %r フィールド記述子に対応します。
- era  
ロケールにおける元号 (アジア各国の日付表記) で、年の計算方法と表示方法を決定する定義。
- era\_d\_fmt  
元号表記での日付だけの書式。 %Ex フィールド記述子に対応します。
- era\_t\_fmt  
元号表記での時刻だけの書式。 %EX フィールド記述子に対応します。
- era\_d\_t\_fmt  
元号表記での日付と時刻の両方を含む書式。 %Ec フィールド記述子に対応します。
- alt\_digits  
アジア各国のロケールで使用される数字の代替シンボルの定義。 %O フィールド記述子に対応します。  
この書式は、日付文字列に代替シンボルが含まれる国で使用します。

他のカテゴリ・セクションと同様に、copy 文を使用して、他のロケールの LC\_TIME 定義全体をインクルードできます。 Tru64 UNIX では、ここで説明したもの以外のシンボルとフィールド記述子もサポートしています。

LC\_TIME 定義についての詳細は、`locale(4)` のリファレンス・ページを参照してください。

## 6.3 マルチバイト・コードとワイド文字コード間の変換を行うライブラリの構築

C ライブラリ・ルーチンは、データ・ファイル・コードとワイド文字コード (内部処理コード) 間の変換を行う特別なインタフェース群を使用します。省略時には、C ライブラリ・ルーチンは、シングルバイト文字だけを扱うインタフェースを使用します。ただし、多くのルーチンには、マルチバイト文字を扱える代替インタフェースを使用するためのエントリ・ポイントが用意されています。ロケールのコードセットに合わせて修正可能なインタフェースはメソッドと呼ばれます。

マルチバイト・コードセットを持つロケールでは、メソッドを使用しなければなりません。また、シングルバイト・コードセットのロケールでも、メソッドを用意しなければならない場合があります。たとえば、ロケールに対応するインタフェースが文字のデータ形式の変換を行い、その操作を正しく実行するために、コードセット固有のロジックが必要になるロケールでは、メソッドを用意する必要があります。ただし、対応するインタフェースがワイド文字に変換済みのデータを扱っており、シングルバイト文字とマルチバイト文字の両方に有効なロジックを使用しているときは、メソッドは無くてもかまいません。

ロケールにメソッドを用意する場合、必要なメソッドのセットを含んでいなければなりません (6.3.1 項を参照)。オプションのメソッドについては、6.3.2 項を参照してください。

メソッドは、システム上ではシェアード・ライブラリとして利用できなければなりません。このライブラリと、各メソッドを実装するライブラリ内の関数は、`methods` ファイルを介して `localedef` コマンドから利用できるようになります。`localedef` コマンドで `methods` ファイルを `charmap` および `locale` ソース・ファイルとともに処理すると、結果として得られるロケールには、ロケールに含まれるすべてのメソッドへのポインタと、ロケールに含まれていない省略時のオプション・メソッドへのポインタが取り込まれます。新たに作成したロケールを `LANG` 変数に設定して、コマンドやアプリケーションを実行すると、メソッドは、システム・ソフトウェアで利用可能な状態になっていれば、常にコマンドやアプリケーションで使用されます。

### 6.3.1 必須メソッド

ロケールでメソッドを使用するときは、次のものが用意されていなければなりません。

- `__mbstopcs` (6.3.1.1 項)
- `__mbtopc` (6.3.1.2 項)
- `__pcstombs` (6.3.1.3 項)
- `__pctomb` (6.3.1.4 項)
- `mblen` (6.3.1.5 項)
- `mbstowcs` (6.3.1.6 項)
- `mbtowc` (6.3.1.7 項)
- `wcstombs` (6.3.1.8 項)
- `wctomb` (6.3.1.9 項)
- `wcswidth` (6.3.1.10 項)
- `wcwidth` (6.3.1.11 項)

これらのメソッドは、C ライブラリ関数でマルチバイト形式とワイド文字形式間のデータ変換を行えるようにします。

#### 6.3.1.1 `fgetws` 関数用の `__mbstopcs` メソッドの作成

`fgetws()` 関数は `__mbstopcs` メソッドを使用して、標準入出力 (stdio) バッファ内のバイトをワイド文字列に変換します。このメソッドを実装する関数は、呼び出しによって変換されたワイド文字の数を返さなければなりません。

このメソッドは `mbstowcs()` (6.3.1.6 項を参照) に類似していますが、`fgetws()` の要件を満たすために、いくつかのパラメータが追加されています。一般に、このメソッドの C ソース・ファイルの名前は、`__mbstopcs_codeset.c` です。`codeset` は、メソッドに対応するコードセットを示します。例 6-10 は、`ja_JP.sdeckanji` ロケールで使用される `__mbstopcs` メソッドを定義するファイル、`__mbstopcs_sdeckanji.c` です。



## 例 6-10: ja\_JP.sdeckanji ロケールのための \_\_mbstopcs\_sdeckanji メソッド

```
#include <stdlib.h> 1
#include <wchar.h> 1
#include <sys/localedef.h> 1

int __mbstopcs_sdeckanji(
    wchar_t *pwcs, 2
    size_t pwcs_len, 3
    const char *s, 4
    size_t s_len, 5
    int stopchr, 6
    char **endpctr, 7
    int *err, 8
    _LC_charmap_t *handle ) 9
{
    int cnt = 0; 10
    int pwcs_cnt = 0; 10
    int s_cnt = 0; 10

    *err = 0; 11

    while (1) { 12
        if (pwcs_cnt >= pwcs_len || s_cnt >= s_len) {
            *endpctr = (char *)&(s[s_cnt]);
            break;
        } 13
        if ((cnt = __mbtopc_sdeckanji(&(pwcs[pwcs_cnt]),
            &(s[s_cnt]), (s_len - s_cnt), err)) == 0) {
            *endpctr = (char *)&(s[s_cnt]);
            break;
        } 14
        pwcs_cnt++; 15
        if (s[s_cnt] == (char) stopchr) {
            *endpctr = (char *)&(s[s_cnt+1]);
            break;
        } 16
        s_cnt += cnt; 17
    } 18
    return (pwcs_cnt); 19
}
```

- 1 このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- 2 ワイド文字列を格納しているバッファを `pwcs` でポイントします。
- 3 `pwcs` バッファのサイズを格納するための変数 `pwcs_len` を定義します。
- 4 変換対象のマルチバイト文字列を格納しているバッファを `s` でポイントします。
- 5 `s` バッファ内のデータのバイト数を格納するための変数 `s_len` を定義します。

`fgetws()` 関数が読み込む標準入力バッファには、ヌルで終わる文字列が含まれていないため、このパラメータが必要になります。

- ⑥ 変換を強制終了するためのバイト値を格納する変数 `stopchr` を定義します。

一般に、この値は `\n` で、呼び出しごとに 1 行の入力を処理する `fgetws()` 関数からの呼び出しでこのメソッドに渡されます。

- ⑦ 変換された最後のバイトの次のバイトをポイントする変数 `endptr` を定義します。

このポインタは、`fgetws()` の次の呼び出しに備え、標準入力バッファ内の開始文字を指定するために必要です。

- ⑧ このメソッドが `mbtopc` メソッドを呼び出したときの実行ステータスを格納している変数を、`err` でポイントします。

- ⑨ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`hdl` でポイントします。

`localedef` コマンドは `_LC_charmap_t` 構造体を作成し、値を格納します。

- ⑩ マルチバイト形式の文字に必要なバイト数を示す変数 (`mbtopc` メソッドが提供する) と、`fgetws()` 関数を使用するバッファ内のバイトまたは文字位置を示す変数を初期化します。

- ⑪ `err` にゼロ (0) を設定し、成功を示します。

- ⑫ マルチバイト文字列を変換する `while` ループを開始します。

- ⑬ ワイド文字データを含むバッファ内にスペースがなくなるか、マルチバイト・データを含むバッファ内にデータがなくなったときに、`endptr` を設定し、ループから抜けます。

- ⑭ `mbtopc` メソッドを呼び出して、文字をマルチバイト形式からワイド文字形式に変換します。

`mbtopc` メソッドが文字の変換に失敗し、エラーを返した場合、ループから抜け、変換できなかった文字の最初のバイトを `endptr` に設定します。

`err` 変数には、次のいずれかの、`mbtopc` メソッド呼び出しのリターン・ステータスが格納されます。

- 0 は成功を示す。

- -1 は無効な文字を示す。
- 0 より大きい値は、マルチバイト文字バッファに残っているバイト数が少ないため、有効な文字を作成できないことを示す。この場合、有効な文字を作成するのに必要なバイト数が返されます。この後、`fgetws()` 関数はバッファを再度充填し、処理をやり直します。

**15** ワイド文字データを含むバッファ内の文字位置をインクリメントします。

**16** マルチバイト・データ内に `stopchr` 文字が現れた場合、`endptr` を `stopchr` 内の次の文字に設定します。

**17** マルチバイト・データを含むバッファ内のバイト位置をインクリメントします。

**18** `while` ループを終了します。

**19** ワイド文字データを含むバッファ内の文字数を返します。

### 6.3.1.2 `getwc()` 関数用の `__mbtopc` メソッドの作成

`getwc()` または `fgetwc()` 関数は、`__mbtopc` メソッドを呼び出して、マルチバイト文字をワイド文字に変換します。このメソッドは、変換対象のマルチバイト文字に含まれるバイト数を返します。このメソッドは `mbtowc` (6.3.1.7 項を参照) 用のメソッドと類似していますが、`getwc()` が必要とするパラメータが追加されています。一般に、このメソッドの C ソース・ファイルの名前は、`__mbtopc_codeset.c` です。`codeset` は、このメソッドに対応するコードセットを示します。例 6-11 は、`ja_JP.sdeckanji` ロケールで使用される `__mbtopc` メソッドを定義しているファイル、`__mbtopc_sdeckanji.c` です。

#### 例 6-11: `ja_JP.sdeckanji` ロケールのための `__mbtopc_sdeckanji` メソッド

```
#include <stdlib.h> 1
#include <wchar.h>
#include <sys/localedef.h>

/*
The algorithm for this conversion is:
s[0] < 0x9f: PC = s[0]
s[0] = 0x8e: PC = s[1] + 0x5f;
s[0] = 0x8f PC = ((s[1] - 0xa1) << 7) | (s[2] - 0xa1)) + 0x303c
s[0] > 0xa1:0xa1 < s[1] < 0xfe
PC = ((s[0] - 0xa1) << 7) | (s[1] - 0xa1)) + 0x15e
0x21 < s[1] < 0x7e
PC = ((s[0] - 0xa1) << 7) | (s[1] - 0x21)) + 0x5f1a
+-----+-----+-----+-----+
| process code | s[0] | s[1] | s[2] |
+-----+-----+-----+-----+
```

例 6-11: ja\_JP.sdeckanji ロケールのための \_\_mbtopc\_sdeckanji メソッド (続き)

0x0000 - 0x009f	0x00-0x9f	--	--	
0x00a0 - 0x00ff	--	--	--	
0x0100 - 0x015d	0x8e	0xa1-0xfe	--	JIS X0201 RH
0x015e - 0x303b	0xa1-0xfe	0xa1-0xfe	--	JIS X0208
0x303c - 0x5f19	0x8f	0xa1-0xfe	0xa1-0xfe	JIS X0212
0x5f1a - 0x8df7	0xa1-0xfe	0x21-0xfe	--	UDC

```

+-----+
*/ 2
int __mbtopc_sdeckanji(
    wchar_t *pwc, 3
    char *ts, 4
    size_t maxlen, 5
    int *err, 6
    _LC_charmap_t *handle ) 7
{
    wchar_t dummy; 8
    unsigned char *s = (unsigned char *)ts; 9
    if (s == NULL)
        return(0); 10
    if (pwc == (wchar_t *)NULL)
        pwc = &dummy; 11
    *err = 0; 12
    if (s[0] <= 0x8d) {
        if (maxlen < 1) {
            *err = 1;
            return(0);
        }
        else {
            *pwc = (wchar_t) s[0];
            return(1);
        }
    } 13
    else if (s[0] == 0x8e) {
        if (maxlen >= 2) {
            if (s[1] >= 0xa1 && s[1] <= 0xfe) {
                *pwc = (wchar_t) (s[1] + 0x5f);
                return(2);
            }
        }
        else {
            *err = 2;
            return(0);
        }
    } 14
    else if (s[0] == 0x8f) {
        if (maxlen >= 3) {
            if ((s[1] >= 0xa1 && s[1] <= 0xfe) &&
                (s[2] >= 0xa1 && s[2] <= 0xfe)) {
                *pwc = (wchar_t) (((s[1] - 0xa1) << 7) |
                    (wchar_t) (s[2] - 0xa1)) + 0x303c;
                return(3);
            }
        }
        else {
            *err = 3;
            return(0);
        }
    } 15
    else if (s[0] <= 0x9f) {

```

### 例 6-11: ja\_JP.sdeckanji ロケールのための \_\_mbtopc\_sdeckanji メソッド (続き)

```
        if (maxlen < 1) {
            *err = 1;
            return(0);
        }
        else {
            *pwc = (wchar_t) s[0];
            return(1);
        }
    } 16
else if (s[0] >= 0xa1 && s[0] <= 0xfe) {
    if (maxlen >= 2) {
        if (s[1] >= 0xa1 && s[1] <= 0xfe) {
            *pwc = (wchar_t) (((s[0] - 0xa1) << 7) |
                               (wchar_t) (s[1] - 0xa1)) + 0x15e;
            return(2);
        } else if (s[1] >= 0x21 && s[1] <= 0x7e) {
            *pwc = (wchar_t) (((s[0] - 0xa1) << 7) |
                               (wchar_t) (s[1] - 0x21)) + 0x5f1a;
            return(2);
        }
    }
    else {
        *err = 2;
        return(0);
    }
} 17
*err = -1;
return(0); 18
}
```

- 1** このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- 2** コードセットでサポートされる各種の文字セットについて、バイト数と有効バイトの組み合わせを判定するためのアルゴリズムを記述しています。  
  
このコードセットは複数の文字セットをサポートしており、各文字セットに含まれる文字の長さは同じです。最初のバイトの値から文字セットがわかり、同時に文字の長さも判定できます。マルチバイト文字を含む文字セットでは、さらに1つまたは複数のバイトを調べ、値の並びが文字を表現しているか、無効なものかどうかを判定する必要があります。
- 3** ワイド文字を格納しているバッファを `pwc` でポイントします。
- 4** 呼び出し元の関数からメソッドに渡されるバイトを格納しているバッファを、`ts` でポイントします。

- 5 マルチバイト・データ内の最大バイト数を含む変数 `maxlen` を宣言します。
- この値は呼び出し元の関数から渡されます。
- 6 実行ステータスを含むバッファを `err` でポイントします。
- 7 このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`handle` でポイントします。
- 8 変数 `dummy` を宣言します。この変数を `pwc` に設定することにより、`pwc` が有効なアドレスをポイントしていることが保証されます。
- 9 `ts` (符号付き文字の配列) を `s` (符号なし文字の配列) にキャストします。
- この操作を行うことにより、配列に整数値が格納されているときに、それらの値をインデックスで参照する際に起きる問題を回避できます。コンパイラは、`char` のような小さな符号付きデータ型を、`int` のような大きな符号付きデータ型と比較するときに、値の符号拡張を行います。この場合、次のような条件は、偽になると期待していても真と評価されます。
- ```
if (s[0] <= 0x8d
```
- 10 `s` バッファが `NULL` を含んでいるか、`NULL` をポイントしているときは、ゼロ (0) を返します。
- 11 `ts` バッファが `NULL` を含んでいるか、`NULL` をポイントしているときは、`dummy` の内容をワイド文字バッファに格納します。
- この操作により、`*pwc` は常に有効なアドレスをポイントします。有効なアドレスをポイントしておらず、ワイド文字が `pwc` に格納されていない場合、アプリケーションがこのポインタを参照すると、セグメンテーション違反が発生します。
- 12 `err` にゼロ (0) を設定し、成功を示します。
- 13 文字が、コードセットで `0x8d` 以下の値として定義されているシングルバイト文字の 1 つであるかどうかを判定します。
- `s` が文字を含んでいない場合、ゼロ (0) を返し、バイトの変換が行われなかったことを示します。また、`err` に 1 を設定し、有効な文字を作成するには 1 バイトが必要であることを示します。
- バイト値がテスト範囲に含まれている場合、対応する処理コード値を `pwc` に移動して、変換したバイトの数を示すために 1 を返します。

- 14 文字が、コードセットで値 0x8e (第 1 バイト) と、0xa1 から 0xfe の範囲の値 (第 2 バイト) として定義されている 2 バイト文字の 1 つであるかどうかを判定します。

文字が条件を満たす場合、対応する処理コード値を `pwc` バッファに移動して、変換したバイトの数を示すために 2 を返します。そうでなければ、ゼロ (0) を返し、変換が行われなかったことを示します。また、`err` に 2 を設定し、有効な文字を作成するには少なくとも 2 バイトが必要であることを示します。

- 15 文字が、コードセットで値 0x8f (第 1 バイト)、0xa1 から 0xfe の範囲の値 (第 2 バイト)、および 0xa1 から 0xfe の範囲の値 (第 3 バイト) として定義されている 3 バイト文字の 1 つであるかどうかを判定します。

文字が条件を満たす場合、対応する処理コード値を `pwc` バッファに移動し、変換したバイトの数を示すために 3 を返します。そうでなければ、ゼロ (0) を返し、変換が行われなかったことを示します。また、`err` に 3 を設定し、有効な文字を作成するには少なくとも 3 バイトが必要であることを示します。

- 16 文字が、コードセットで 0x90 から 0x9f の範囲の値として定義されているシングルバイト文字の 1 つであるかどうかを判定します。

標準入出力バッファにバイト・データがない場合、ゼロ (0) を返し、変換が行われなかったことを示します。また、`err` に 1 を設定し、有効な文字を作成するには少なくとも 1 バイトが必要であることを示します。

バイト値が定義された範囲に含まれていれば、対応する処理コード値を `pwc` に移動し、変換したバイトの数を示すために 1 を返します。

- 17 文字が、コードセットで 0xa1 から 0xfe の範囲の値 (第 1 バイト) と、0x21 から 0x7e の範囲の値 (第 2 バイト) として定義されている 2 バイト文字の 1 つであるかどうかを判定します。

文字が条件を満たす場合、対応する処理コード値を `pwc` バッファに移動し、変換したバイトの数を示すために 2 を返します。そうでなければ、ゼロ (0) を返し、変換が行われなかったことを示します。また、`err` に 2 を設定し、有効な文字を作成するには少なくとも 2 バイトが必要であることを示します。

- 18 `err` に -1 を設定し、無効なマルチバイト・シーケンスが存在していたことを示します。また、ゼロ (0) を返し、変換が行われなかったことを示します。

これらの文は、`s` 中のマルチバイト・データが、上記の `if` 条件をいずれも満たしていない場合に実行されます。

### 6.3.1.3 `fputws()` 関数用の `__pcstombs` メソッドの作成

`fputws()` 関数は `__pcstombs` メソッドを呼び出して、文字列を処理 (ワイド文字) コードからマルチバイト・コードに変換します。このメソッドが `-1` を返し、ロケールでサポートされていないことを示した場合、`fputws()` は、変換する文字列中のワイド文字ごとに `putwc()` を呼び出します。一般に、このメソッドの C ソース・ファイルの名前は、`__pcstombs_codeset.c` です。`codeset` は、このメソッドに対応するコードセットを示します。例 6-12 は、`ja_JP.sdeckanji` ロケールで使用する `__pcstombs` メソッドを定義するファイル、`__pcstombs_sdeckanji.c` です。

例 6-12: `ja_JP.sdeckanji` ロケールのための `__pcstombs_sdeckanji` メソッド

---

```
int __pcstombs_sdeckanji()
{
    return -1; ❶
}
```

---

❶ `-1` を返し、このメソッドがロケールでサポートされていないことを示します。

この戻り値により、`fputws()` 関数は `putwc()` を複数回呼び出して、文字列中のワイド文字を変換します。

単に `-1` を返すだけではない完全なメソッドを実装したい場合には、関数が変換されたワイド文字の数を返すようにしなければなりません。また、次の例に示すヘッダ・ファイルとパラメータをインクルードする必要があります。

```
#include <stdlib.h>
#include <wchar.h>
#include <sys/localedef.h>

int __pcstombs_newcodeset(
    wchar_t *pcsbuf, ❶
    size_t pcsbuf_len, ❷
    char *mbsbuf, ❸
    size_t mbsbuf_len, ❹
    char **endptr, ❺
    int *err, ❻)
```



`_LC_charmap_t *handle )` **7**

**1** ワイド文字列を含むバッファへのポインタを指定します。

**2** ワイド文字バッファの大きさを含む変数を指定します。

この値は、`fputws()` からの呼び出しでメソッドに渡されます。

**3** マルチバイト文字列を含むバッファへのポインタを指定します。

**4** マルチバイト文字バッファの大きさを含む変数を指定します。

この値は、`fputws()` からの呼び出しでメソッドに渡されます。

**5** すべてのワイド文字データを変換するために `fputws()` を複数回呼び出さなければならない場合、マルチバイト文字バッファ内の次の文字の開始バイト位置へのポインタを、`endptr` でポイントします。

**6** 実行ステータスを返すためのポインタを指定します。

このメソッドが文字変換を行うために `wctomb` メソッドを呼び出す場合、このステータスは `wctomb` が設定します。それ以外の場合には、このメソッドはワイド文字からマルチバイト文字への変換を実行するロジックを備えている必要があり、また、このメソッドがステータスを直接設定しなければなりません。

いずれの場合でも、`fputws()` 関数は次の値を想定します。

- 成功時には 0。
- ワイド文字の値が無効なため変換できない場合は -1。
- 最後の文字を処理した後で、マルチバイト文字バッファ内のバイト数が新しい文字を格納するのに足りない場合は正の値。

この場合、値は次の文字を格納するのに必要なバイト数です。この後、`fputws()` 関数はマルチバイト文字バッファを空にして、再試行します。

**7** このロケールで使用されるメソッドへのポインタを格納している

`_LC_charmap_t` 構造体へのポインタを指定します。

`__pcstombs` メソッドは、`__mbstopcs` メソッドが実行する操作 (6.3.1.1 項を参照) の逆の操作を行います。データ変換の向きが異なるため、`__pcstombs` メソッドには次のような特徴があります。

- `\n` などの変換停止文字のための変数を必要としない。

- `mbtowc` メソッドではなく `wctomb` メソッドを呼び出して (あるいは, `wctomb` メソッドが行う操作を実装することにより), 個々の文字を変換し, マルチバイト文字バッファ内でこのメソッドが必要とするバイト数を決定する。

#### 6.3.1.4 `__pctomb` メソッドの作成

C ライブラリ関数は, 現時点では `__pctomb` インタフェースを使用しません。たとえば `putwc()` 関数は, `wctomb` メソッドを呼び出して, 文字をワイド文字形式からマルチバイト文字形式に変換します。ただし, `localedef` コマンドは, ロケールでメソッドが提供されている場合には, この関数のためのメソッドを必要とします。一般に, このメソッドの C ソース・ファイルの名前は, `__pctomb_codeset.c` です。 `codeset` は, このメソッドに対応するコードセットを示します。例 6-13 は, `ja_JP.sdeckanji` ロケールで使用される `__pctomb` メソッドを定義するファイル, `__pctomb_sdeckanji.c` です。

例 6-13: `ja_JP.sdeckanji` ロケールのための `__pctomb_sdeckanji` メソッド

---

```
int __pctomb_sdeckanji()
{
    return -1; ❶
}
```

---

- ❶ `-1` を返し, ロケールでこのメソッドがサポートされていないことを示します。

#### 6.3.1.5 `mblen()` 関数用のメソッドの作成

`mblen()` 関数は `mblen` メソッドを使用して, マルチバイト文字のバイト数を返します。一般に, このメソッドの C ソース・ファイルの名前は, `__mblen_codeset.c` です。 `codeset` は, このメソッドに対応するコードセットを示します。例 6-14 は, `ja_JP.sdeckanji` ロケールで使用される `mblen` メソッドを定義するファイル, `__mblen_sdeckanji.c` です。

## 例 6-14: ja\_JP.sdeckanji ロケールのための \_\_mblen\_sdeckanji メソッド

```
#include <stdlib.h> [1]
#include <wchar.h>
#include <sys/errno.h>
#include <sys/localedef.h>

/*
The algorithm for this conversion is:

s[0] < 0x9f: 1 byte
s[0] = 0x8e: 2 bytes
s[0] = 0x8f: 3 bytes
s[0] > 0xa1: 2 bytes

+-----+-----+-----+-----+
| process code | s[0] | s[1] | s[2] |
+-----+-----+-----+-----+
| 0x0000 - 0x009f | 0x00-0x9f | -- | -- |
| 0x00a0 - 0x00ff | -- | -- | -- |
| 0x0100 - 0x015d | 0x8e | 0xa1-0xfe | -- | JIS X0201 RH
| 0x015e - 0x303b | 0xa1-0xfe | 0xa1-0xfe | -- | JIS X0208
| 0x303c - 0x5f19 | 0x8f | 0xa1-0xfe | 0xa1-0xfe | JIS X0212
| 0x5f1a - 0x8df7 | 0xa1-0xfe | 0x21-0xfe | -- | UDC
+-----+-----+-----+-----+

*/ [2]

int __mblen_sdeckanji(
    char *fs, [3]
    size_t maxlen, [4]
    _LC_charmap_t *handle ) [5]
{
    const unsigned char *s = (void *) fs; [6]

    if (s == NULL || *s == '\0')
        return(0); [7]

    if (maxlen < 1) {
        _Seterrno(EILSEQ);
        return((size_t)-1);
    } [8]

    if (s[0] <= 0x8d)
        return(1); [9]

    else if (s[0] == 0x8e) {
        if (maxlen >= 2 && s[1] >=0xa1 && s[1] <=0xfe)
            return(2);
    } [10]

    else if (s[0] == 0x8f) {
        if(maxlen >=3 && (s[1] >=0xa1 && s[1] <=0xfe) &&
            (s[2] >=0xa1 && s[2] <= 0xfe))
            return(3);
    } [11]

    else if (s[0] <= 0x9f)
        return(1); [12]

    else if (s[0] >= 0xa1) {
        if (maxlen >=2 && (s[0] <= 0xfe) )
            if ( (s[1] >=0xa1 && s[1] <= 0xfe) ||
                (s[1] >=0x21 && s[1] <= 0x7e) )
                return(2);
    }
}
```

例 6-14: ja\_JP.sdeckanji ロケールのための `__mblen_sdeckanji` メソッド (続き)

```
    } 13  
  
    _Seterrno(EILSEQ);  
    return((size_t)-1); 14  
}
```

- ❶ このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- ❷ 文字のバイト数と、その文字が有効なバイト・シーケンスであるかどうかを決定するためのアルゴリズムを記述します。  
  
このコードセットは複数の文字セットをサポートしており、各文字に含まれる文字の長さは同じです。最初のバイトの値から文字セットがわかり、同時に文字の長さも判定できます。マルチバイト文字を含む文字セットでは、さらに1つまたは複数のバイトを調べ、値の並びが文字を表現しているか、無効なものかどうかを判定する必要があります。
- ❸ 検査するバイト文字列を含むバッファを `fs` でポイントします。
- ❹ マルチバイト文字の最大長を含む変数 `maxlen` を定義します。  
この値は、`mblen()` 関数からこのメソッドに渡されます。
- ❺ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`handle` でポイントします。
- ❻ `fs` (符号付き文字の配列) を `s` (符号なし文字の配列) にキャストします。  
  
この操作を行うことにより、配列に整数値が格納されているときに、インデックスで参照する際に起きる問題を回避できます。コンパイラは、`char` のような小さな符号付きデータ型を、`int` のような大きな符号付きデータ型と比較するときに、値の符号拡張を行います。この場合、次のような条件は、偽になると期待していても真と評価されます。  
  

```
if (s[0] <= 0x8d
```
- ❼ `s` が `NULL` を含んでいるか、`NULL` をポイントしているときは、ゼロ (0) を返し、文字の長さがゼロ (0) であることを示します。

- ⑧ maxlen (最大バイト数) がゼロか、負の値の場合には、-1 を返し、`errno` に [EILSEQ] (無効な文字シーケンス) を設定します。

マルチスレッド・アプリケーションで `errno` が正しく機能するように設定するには、代入文ではなく `_seterrno` を使用します。

- ⑨ 最初のバイトが 0x8d 以下のシングルバイト文字を示しているかどうかを判定します。

そうであれば、1 を返して、文字の長さが 1 バイトであることを示します。

- ⑩ 最初のバイトが値 0x8e を含み、2 番目のバイトが 0xa1 から 0xfe の範囲の値を含む 2 バイト文字を示しているかどうかを判定します。

そうであれば、2 を返して、文字の長さが 2 バイトであることを示します。

- ⑪ 最初のバイトが値 0x8f を含み、2 番目と 3 番目のバイトが 0xa1 から 0xfe の範囲の値を含む 3 バイト文字を示しているかどうかを判定します。

そうであれば、3 を返して、文字の長さが 3 バイトであることを示します。

- ⑫ 最初のバイトが 0x9f 以下のシングルバイト文字を示しているかどうかを判定します。

そうであれば、1 を返して、文字の長さが 1 バイトであることを示します。

- ⑬ 最初のバイトが 0xa1 から 0xfe の範囲の値を含み、2 番目のバイトが 0x21 から 0x7e の範囲の値を含む 2 バイト文字を示しているかどうかを判定します。

そうであれば、2 を返して、文字の長さが 2 バイトであることを示します。

- ⑭ -1 を返し、`errno` に [EILSEQ] を設定して、無効なマルチバイト・シーケンスであることを示します。

これらの文は、標準入出力バッファ内のマルチバイト・データが、上記の `if` 条件をいずれも満たさない場合に実行されます。

### 6.3.1.6 mbstowcs() 関数用のメソッドの作成

mbstowcs() 関数は mbstowcs メソッドを使用して、マルチバイト文字列をプロセス・ワイド文字コードに変換し、結果として得られたワイド文字の数を返します。一般に、このメソッドの C ソース・ファイルの名前は、\_\_mbstowcs\_codeset.c です。codeset は、このメソッドに対応するコードセットを示します。例 6-15 は、ja\_JP.sdeckanji ロケールで使用される mbstowcs メソッドを定義するファイル、\_\_mbstowcs\_sdeckanji.c です。

#### 例 6-15: ja\_JP.sdeckanji ロケールのための \_\_mbstowcs\_sdeckanji メソッド

```
#include <stdlib.h> 1
#include <wchar.h>
#include <sys/localedef.h>

size_t __mbstowcs_sdeckanji(
    wchar_t *pwcs, 2
    const char *s, 3
    size_t n, 4
    _LC_uchar_t *handle ) 5
{
    int len = n; 6
    int rc; 7
    int cnt; 8
    wchar_t *pwcs0 = pwcs; 9
    int mb_cur_max; 10

    if (s == NULL)
        return (0); 11

    mb_cur_max = MB_CUR_MAX; 12

    if (pwcs == (wchar_t *)NULL) {
        cnt = 0;
        while (*s != '\0') {
            if ((rc = __mblen_sdeckanji(s, mb_cur_max, handle)) == -1)
                return(-1);
            cnt++;
            s += rc;
        }
        return(cnt);
    } 13

    while (len-- > 0) {
        if (*s == '\0') {
            *pwcs = (wchar_t) '\0';
            return (pwcs - pwcs0);
        }
        if ((cnt = __mbtowc_sdeckanji(pwcs, s, mb_cur_max, handle)) < 0)
            return(-1);
        s += cnt;
        ++pwcs;
    } 14

    return (n); 15
```

例 6-15: ja\_JP.sdeckanji ロケールのための \_\_mbstowcs\_sdeckanji メソッド (続き)

---

```
}
```

---

- ❶ このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- ❷ ワイド文字列を格納しているバッファを `pwcs` でポイントします。
- ❸ マルチバイト文字列を格納しているバッファを `s` でポイントします。
- ❹ `pwcs` 中のワイド文字数を含む変数 `n` を定義します。
- ❺ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を `handle` でポイントします。
- ❻ `pwcs` バッファ内のワイド文字の数 (呼び出し元関数が指定した `n` の値) を `len` に代入します。
- ❼ このメソッドから呼び出される `mblen` 関数が返す値を含む変数 `rc` を定義します。
- ❽ `s` バッファ内の文字に使用されるバイト数をカウントする変数 `cnt` を定義します。
- ❾ 呼び出し元の関数が `pwcs0` 変数に渡すワイド文字列の開始点を保存します。
- ❿ 変数 `mb_cur_max` を定義します。この変数は後で `MB_CUR_MAX` に設定され、`mblen` メソッドの呼び出しで使用されます。
- ⓫ `s` が `NULL` の場合、ゼロ (0) を返します。  
メソッドは、ロケールにおける文字のエンコーディングに状態がなければゼロ (0) を返し、状態があればゼロ以外の値を返します。
- ⓬ `MB_CUR_MAX` に設定されている値を `mb_cur_max` に代入し、後で `mblen` メソッドの呼び出しに使用します。
- ⓭ `NULL` ポインタが呼び出し元の関数から渡されたかどうかを調べ、渡されていれば `mblen` メソッドを呼び出してワイド文字列のサイズを計算します。

`mbstowcs()` の呼び出しでヌル・ワイド文字を `pwcs` パラメータとして渡すことにより、メモリ割り当てに使用する `pwcs` バッファのサイズを取得できます。また、この戻り値を使用することにより、再度 `mbstowcs()` を呼び出してマルチバイト文字列を実際に変換する前に、アプリケーションのワイド文字バッファのメモリ・スペースを効率的に割り当てることができます。

- 14 ヌル文字 (文字列の終わり) が現れるまで `__mbtowc` メソッドを呼び出すことにより、マルチバイト文字バッファ内のバイトを変換します。

`NULL` が検出されたら処理を停止し、`pwcs` バッファ内のワイド文字の数を返します。文字の変換に成功するたびに、マルチバイト文字バッファ内のバイト位置を適切な数だけインクリメントします。

この `while` ループは、条件 `len-- > 0` を使用して、`pwcs` バッファが一杯になったときに処理を停止します。ループ中の最初の `if` 条件は、`s` バッファ内のマルチバイト文字列がヌルで終了しているときに、`pwcs` バッファ内の対応するヌル終端子が、`mbtowcs()` 関数によりアプリケーションに返されるワイド文字数に含まれないようにします。

- 15 `n` の値を返して、`pwcs` バッファ内のワイド文字の数を示します。

この文は、`s` バッファ内でヌルが検出される前に、`pwcs` バッファのスペースが足りなくなった場合に実行されます。

### 6.3.1.7 `mbtowc()` 関数用のメソッドの作成

`mbtowc()` 関数は `mbtowc` メソッドを使用して、マルチバイト文字をワイド文字に変換し、変換したマルチバイト文字のバイト数を返します。一般に、このメソッドの C ソース・ファイルの名前は、`__mbtowc_codeset.c` です。`codeset` は、このメソッドが対応するコードセットを示します。例 6-16 は、`ja_JP.sdeckanji` ロケールで使用される `mbtowc` メソッドを定義するファイル、`__mbtowc_sdeckanji.c` です。

#### 例 6-16: `ja_JP.sdeckanji` ロケールのための `__mbtowc_sdeckanji` メソッド

```
#include <stdlib.h> 1
#include <wchar.h>
#include <sys/errno.h>
#include <sys/localedef.h>

/*
The algorithm for this conversion is:

s[0] < 0x9f: PC = s[0]
s[0] = 0x8e: PC = s[1] + 0x5f;
```



## 例 6-16: ja\_JP.sdeckanji ロケールのための \_\_mbtowc\_sdeckanji メソッド (続き)

```
s[0] = 0x8f    PC = (((s[1] - 0xa1) << 7) | (s[2] - 0xa1)) + 0x303c
s[0] > 0xa1:0xa1 < s[1] < 0xfe
                PC = (((s[0] - 0xa1) << 7) | (s[1] - 0xa1)) + 0x15e
0x21 < s[1] < 0x7e
                PC = (((s[0] - 0xa1) << 7) | (s[1] - 0x21)) + 0x5f1a
```

| process code    | s[0]      | s[1]      | s[2]      |              |
|-----------------|-----------|-----------|-----------|--------------|
| 0x0000 - 0x009f | 0x00-0x9f | --        | --        |              |
| 0x00a0 - 0x00ff | --        | --        | --        |              |
| 0x0100 - 0x015d | 0x8e      | 0xa1-0xfe | --        | JIS X0201 RH |
| 0x015e - 0x303b | 0xa1-0xfe | 0xa1-0xfe | --        | JIS X0208    |
| 0x303c - 0x5f19 | 0x8f      | 0xa1-0xfe | 0xa1-0xfe | JIS X0212    |
| 0x5f1a - 0x8df7 | 0xa1-0xfe | 0x21-0xfe | --        | UDC          |

```
*/ 2
int __mbtowc_sdeckanji(
    wchar_t *pwc, 3
    const char *ts, 4
    size_t maxlen, 5
    _LC_charmap_t *handle ) 6
{
    unsigned char *s = (unsigned char *)ts; 7
    wchar_t dummy; 8

    if (s == NULL)
        return(0); 9

    if (maxlen < 1) {
        _Seterrno(EILSEQ);
        return((size_t)-1);
    } 10

    if (pwc == (wchar_t *)NULL)
        pwc = &dummy; 11

    if (s[0] <= 0x8d) {
        *pwc = (wchar_t) s[0];
        if (s[0] != '\0')
            return(1);
        else
            return(0);
    } 12

    else if (s[0] == 0x8e) {
        if ((maxlen >= 2) && ((s[1] >=0xa1) && (s[1] <=0xfe))) {
            *pwc = (wchar_t) (s[1] + 0x5f); /* 0x100 - 0xa1 */
            return(2);
        }
    } 13

    else if (s[0] == 0x8f) {
        if((maxlen >= 3) && (((s[1] >=0xa1) && (s[1] <=0xfe))
            && ((s[2] >=0xa1) && (s[2] <= 0xfe)))) {
            *pwc = (wchar_t) (((s[1] - 0xa1) << 7) |
                (wchar_t) (s[2] - 0xa1)) + 0x303c;
            return(3);
        }
    }
}
```

## 例 6-16: ja\_JP.sdeckanji ロケールのための \_\_mbtowc\_sdeckanji メソッド (続き)

```
    } 14

    else if (s[0] <= 0x9f) {
        *pwc = (wchar_t) s[0];
        if (s[0] != '\0')
            return(1);
        else
            return(0);
    } 15

    else if (((s[0] >= 0xa1) && (s[0] <= 0xfe)) && (maxlen >= 2)){
        if (((s[1] >= 0xa1) && (s[1] <= 0xfe))){
            *pwc = (wchar_t) (((s[0] - 0xa1) << 7) |
                               (wchar_t)(s[1] - 0xa1)) + 0x15e;
            return(2);
        } else if (((s[1] >= 0x21) && (s[1] <= 0x7e))){
            *pwc = (wchar_t) (((s[0] - 0xa1) << 7) |
                               (wchar_t)(s[1] - 0x21)) + 0x5f1a;
            return(2);
        }
    } 16
    _Seterrno(EILSEQ);
    return(-1); 17
}
```

- ❶ このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- ❷ 文字のバイト数と、その文字が有効なバイト・シーケンスであるかどうかを判定するためのアルゴリズムを記述します。

このコードセットは複数の文字セットをサポートしており、各文字セットに含まれる文字の長さは同じです。最初のバイトの値から文字セットがわかり、同時に文字の長さも判定できます。マルチバイト文字を含む文字セットでは、さらに1つまたは複数のバイトを調べ、値の並びが文字を表現しているか、無効なものかどうかを判定する必要があります。

- ❸ ワイド文字を含むバッファを `pwc` でポイントします。
- ❹ マルチバイト文字形式の値を含むバッファを `ts` でポイントします。
- ❺ マルチバイト文字の最大長を含む変数 `maxlen` を定義します。

この値は呼び出し元の関数から渡されます。値は、アプリケーション・プログラマが最初に行った呼び出しで、`MB_CUR_MAX` に設定されています。

- ⑥ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`handle` でポイントします。

- ⑦ `ts` (符号付き文字の配列) を `s` (符号なし文字の配列) にキャストします。  
この操作を行うことにより、配列に整数値が格納されているときに、インデックスで参照する際に起きる問題を回避できます。コンパイラは、`char` のような小さな符号付きデータ型を、`int` のような大きな符号付きデータ型と比較するときに、値の符号拡張を行います。この場合、次のような条件は、偽になると期待していても真と評価されます。

```
if (s[0] <= 0x8d
```

- ⑧ 変数 `dummy` を宣言します。この変数を `pwc` に設定することにより、`pwc` が有効なアドレスをポイントすることが保証されます。
- ⑨ `s` が `NULL` を含んでいるか、`NULL` をポイントしているときは、ゼロ (0) を返して、ロケールにおける文字のエンコーディングに状態がないことを示します。

`NULL` ポインタが渡された場合、このメソッドは、ロケールにおける文字のエンコーディングに状態があるかどうかを示す値を返さなければなりません。状態がある場合は、ゼロ以外の値を返します。

- ⑩ マルチバイト・データ・バッファの長さが 1 バイト未満であれば、`size_t` にキャストされた `-1` を返し、`errno` に `[EILSEQ]` (無効なバイト・シーケンス) を設定します。

- ⑪ `ts` バッファが `NULL` を含んでいるか、`NULL` をポイントしているときは、`dummy` の内容をワイド文字バッファに格納します。

この操作により、`pwc` は常に有効なアドレスをポイントします。この操作を行わないと、ワイド文字が `pwc` に格納されていないときにアプリケーションがこのポインタを参照した場合、セグメンテーション違反が発生します。

- ⑫ 最初のバイトが、`0x8d` 以下のシングルバイト文字を示しているかどうかを判定します。

そうであれば、対応する処理コード値を `pwc` バッファに格納し、文字の長さが 1 バイトであることを示す `1` を返します。

- ⑬ 最初のバイトが値 `0x8e` を含み、2 番目のバイトが `0xa1` から `0xfe` の範囲の値を含む 2 バイト文字を示しているかどうかを判定します。

そうであれば、対応する処理コード値を `pwc` バッファに格納し、文字の長さが 2 バイトであることを示す 2 を返します。

- [14] 最初のバイトが値 `0x8f` を含み、2 番目と 3 番目のバイトが `0xa1` から `0xfe` の範囲の値を含む 3 バイト文字を示しているかどうかを判定します。

そうであれば、対応する処理コード値を `pwc` バッファに格納し、文字の長さが 3 バイトであることを示す 3 を返します。

- [15] 最初のバイトの値が `0x9f` 以下のシングルバイト文字を示しているかどうかを判定します。

そうであれば、対応する処理コード値を `pwc` バッファに格納し、文字の長さが 1 バイトであることを示す 1 を返します。

- [16] 最初のバイトが `0xa1` から `0xfe` の範囲の値を含み、2 番目のバイトが `0x21` から `0x7e` の範囲の値を含む 2 バイト文字を示しているかどうかを判定します。

そうであれば、対応する処理コード値を `pwc` バッファに格納し、文字の長さが 2 バイトであることを示す 2 を返します。

- [17] `-1` を返し、`errno` に `[EILSEQ]` を設定して、無効なマルチバイト・シーケンスであることを示します。

これらの文は、`s` バッファ内のマルチバイト・データが上記の `if` 条件をいずれも満たさない場合に実行されます。

#### 6.3.1.8 `wcstombs()` 関数用のメソッドの作成

`wcstombs()` 関数は `wcstombs` メソッドを呼び出して、ワイド文字列をマルチバイト文字列に変換し、結果として得られたマルチバイト文字列のバイト数を返します。一般に、このメソッドの C ソース・ファイルの名前は、`__wcstombs_codeset.c` です。`codeset` は、このメソッドに対応するコードセットを示します。例 6-17 は、`ja_JP.sdeckanji` ロケールで 사용되는 `wcstombs` メソッドを定義するファイル、`__wcstombs_sdeckanji.c` です。

### 例 6-17: ja\_JP.sdeckanji ロケールのための \_\_wcstombs\_sdeckanji メソッド

```
#include <stdlib.h> 1
#include <wchar.h>
#include <limits.h>
#include <sys/localedef.h>

size_t __wcstombs_sdeckanji(
    char *s, 2
    const wchar_t *pwcs, 3
    size_t n, 4
    _LC_charmap_t *handle ) 5
{
    int cnt=0; 6
    int len=0; 7
    int i=0; 8
    char tmps[MB_LEN_MAX+1]; 9

    if ( s == (char *)NULL ) {
        cnt = 0;
        while (*pwcs != (wchar_t)'\0') {
            if ((len = __wctomb_sdeckanji(tmps, *pwcs)) == -1)
                return(-1);
            cnt += len;
            pwcs++;
        }
        return(cnt);
    } 10

    if (*pwcs == (wchar_t)'\0') {
        *s = '\0';
        return(0);
    } 11

    while (1) { 12

        if ((len = __wctomb_sdeckanji(tmps, *pwcs)) == -1)
            return(-1); 13

        else if (cnt+len > n) {
            *s = '\0';
            break;
        } 14

        if (tmps[0] == '\0') {
            *s = '\0';
            break;
        } 15

        for (i=0; i<len; i++) {
            *s = tmps[i];
            s++;
        } 16

        cnt += len; 17

        if (cnt == n)
            break; 18

        pwcs++; 19
    } 20
```

例 6-17: ja\_JP.sdeckanji ロケールのための \_\_wcstombs\_sdeckanji メソッド (続き)

---

```
if (cnt == 0)
    cnt = len; 21
return (cnt); 22
}
```

---

- ❶ このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- ❷ このメソッドが呼び出し元の関数に渡すマルチバイト文字列を格納しているバッファを, `s` でポイントします。
- ❸ 変換するワイド文字列を格納しているバッファを `pwcs` でポイントします。
- ❹ マルチバイト文字列バッファ内の最大バイト数を格納する変数 `n` を定義します。  
この値は, 呼び出し元の関数が指定します。
- ❺ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を, `handle` でポイントします。
- ❻ 変換された文字のバイト数 (`len`) でインクリメントされる変数 `cnt` を初期化します。
- ❼ 変換された文字の長さを含む変数 `len` を初期化します。
- ❽ 変換した文字を一時的な記憶領域から `s` に移動するときに, 各マルチバイト文字中のバイトに対するインデックスとして使用する変数 `i` を初期化します。
- ❾ `wctomb` メソッドの呼び出しで返されるマルチバイト文字を格納している一時的なバッファ `tmps` を定義します。
- ❿ 呼び出し元の関数が `s` バッファに `NULL` を渡したかどうかを調べます。  
`NULL` が渡されている場合, `wctomb` メソッドを呼び出して, マルチバイト文字バッファ内の変換された文字に必要なバイト数 (ヌル終端子は除く) を計算します。

`wcstombs()` の呼び出しでヌル・バイトを `s` パラメータとして渡すことにより, メモリ割り当てに使用する `s` バッファのサイズを取得できます。

また、この戻り値を使用することにより、再度 `wcstombs()` を呼び出してワイド文字列を実際に変換する前に、アプリケーションのワイド文字バッファのメモリ・スペースを効率的に割り当てることができます。

- [11] ゼロ (0) を返し、マルチバイト文字が得られなかったことを示します。また、`pwcs` が `NULL` をポイントしていれば、`s` に `NULL` を設定します。
- [12] ワイド文字列中の文字を処理する `while` ループを開始します。
- [13] `wctomb` メソッドを呼び出して、ワイド文字バッファ内の文字を変換します。`wctomb` が `-1` を返したときは、`-1` を返して、文字が無効であることを示します。
- [14] `wctomb` が変換した文字のためのスペースが `s` になれば、`s` の末尾に `NULL` を置き、`while` ループを抜けます。
- [15] `s` 中で `NULL` が検出されときは、`s` にヌル終端子を移動して、`while` ループを抜けます。
- [16] 現在のワイド文字が `NULL` でない場合、`tmps` 内の各バイトを `s` に追加します。
- [17] マルチバイト形式でこの文字が占めるバイト数 (`len`) だけ、`cnt` をインクリメントします。
- [18] 処理されたバイト数が `n` (`s` の最大バイト数) に等しいときは、ヌル終端子を追加せずに `while` ループを抜けます。
- [19] `pwcs` をインクリメントし、変換する次のワイド文字をポイントするようにします。
- [20] 個々のワイド文字を変換する `while` ループを終了します。
- [21] `s` に 1 文字分のスペースがない場合は、ゼロ (0) を返すようにします。
- [22] 結果として得られたマルチバイト文字列中のバイト数を返します。

#### 6.3.1.9 `wctomb()` 関数用のメソッドの作成

`wctomb()` 関数は `wctomb` メソッドを呼び出して、ワイド文字をマルチバイト文字に変換し、結果として得られたマルチバイト文字のバイト数を返します。一般に、このメソッドの C ソース・ファイルの名前は、`__wctomb_codeset.c` です。`codeset` は、このメソッドに対応するコードセットを示します。例 6-18 は、`ja_JP.sdeckanji` ロケールのための `wctomb` メソッドを定義するファイル、`__wctomb_sdeckanji.c` です。

## 例 6-18: ja\_JP.sdeckanji ロケールのための \_\_wctomb\_sdeckanji メソッド

```
#include <stdlib.h> 1
#include <wchar.h>
#include <sys/errno.h>
#include <sys/localedef.h>

/*
   The algorithm for this conversion is:

PC <= 0x009f:          s[0] = PC
PC >= 0x0100 and PC <=0x015d: s[0] = 0x8e
                           s[1] = PC - 0x005f
PC >= 0x015e and PC <=0x303b: s[0] = ((PC - 0x015e) >> 7) + 0x00a1
                           s[1] = ((PC - 0x015e) & 0x007f) + 0x00a1
PC >= 0x303c and PC <=0x5f19: s[0] = 0x8f
                           s[1] = ((PC - 0x303c) >> 7) + 0x00a1
                           s[2] = ((PC - 0x303c) & 0x007f) + 0x00a1
PC >= 0x5f1a and PC <=0x8df7 s[0] = ((PC - 0x5f1a) >> 7) + 0x00a1
                           s[1] = ((PC - 0x5f1a) & 0x007f) + 0x0021

+-----+-----+-----+-----+
| process code | s[0] | s[1] | s[2] |
+-----+-----+-----+-----+
| 0x0000 - 0x009f | 0x00-0x9f | -- | -- |
| 0x00a0 - 0x00ff | -- | -- | -- |
| 0x0100 - 0x015d | 0x8e | 0xa1-0xfe | -- | JIS X0201 RH
| 0x015e - 0x303b | 0xa1-0xfe | 0xa1-0xfe | -- | JIS X0208
| 0x303c - 0x5f19 | 0x8f | 0xa1-0xfe | 0xa1-0xfe | JIS X0212
| 0x5f1a - 0x8df7 | 0xa1-0xfe | 0x21-0xfe | -- | UDC
+-----+-----+-----+-----+

*/ 2

int __wctomb_sdeckanji(
    char *s, 3
    wchar_t wc, 4
    _LC_charmap_t *handle ) 5
{
    if (s == (char *)NULL)
        return(0); 6

    if (wc <= 0x9f) {
        s[0] = (char) wc;
        return(1);
    } 7

    else if ((wc >= 0x0100) && (wc <= 0x015d)) {
        s[0] = 0x8e;
        s[1] = wc - 0x5f;
        return(2);
    } 8

    else if ((wc >=0x015e) && (wc <= 0x303b)) {
        s[0] = (char) (((wc - 0x015e) >> 7) + 0x00a1);
        s[1] = (char) (((wc - 0x015e) & 0x007f) + 0x00a1);
        return(2);
    } 9

    else if ((wc >=0x303c) && (wc <= 0x5f19)) {
        s[0] = 0x8f;
        s[1] = (char) (((wc - 0x303c) >> 7) + 0x00a1);
        s[2] = (char) (((wc - 0x303c) & 0x007f) + 0x00a1);
        return(3);
    } 10
```



例 6-18: ja\_JP.sdeckanji ロケールのための \_\_wctomb\_sdeckanji メソッド (続き)

---

```
else if ((wc >= 0x5f1a) && (wc <= 0x8df7)) {
    s[0] = (char) (((wc - 0x5f1a) >> 7) + 0x00a1);
    s[1] = (char) (((wc - 0x5f1a) & 0x007f) + 0x0021);
    return(2);
} 11

_Seterrno(EILSEQ);
return(-1); 12
}
```

---

- 1 このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- 2 このメソッドが使用する変換アルゴリズムを記述します。

コードセットでサポートされる各文字セットは、ワイド文字 (処理コード) 値の一意の範囲に対応しています。個々の文字セット中では、マルチバイト文字の長さは一定 (1, 2 または 3 バイト) です。そのため、個々のワイド文字が属している範囲は、マルチバイト形式の文字に必要なバイト数を示します。ワイド文字値そのものは、その文字のマルチバイト形式における特定の値を決定します。
- 3 マルチバイト文字を含むバッファを `s` でポイントします。
- 4 ワイド文字を含む変数 `wc` を定義します。
- 5 このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`handle` でポイントします。
- 6 `s` が `NULL` をポイントするときは、ゼロ (0) を返して、文字の変換が行われなかったことを示します。
- 7 ワイド文字値が `0x9f` 以下の場合、その値を配列 `s` の最初のバイトに移動して、変換された文字の長さが 1 バイトであることを示す 1 を返します。
- 8 ワイド文字値が `0x0100` から `0x015d` の範囲にあれば、値 `0x8e` を配列 `s` の最初のバイトに移動し、計算した値を 2 番目のバイトに移動します。2 を返して、変換された文字の長さが 2 バイトであることを示します。

- ⑨ ワイド文字値が 0x015e から 0x303b の範囲にあれば、計算した値を配列 *s* の最初のバイトと 2 番目のバイトに移動します。2 を返して、変換された文字の長さが 2 バイトであることを示します。
  - ⑩ ワイド文字値が 0x303c から 0x5f19 の範囲にあれば、値 0x8f を配列 *s* の最初のバイトに移動し、計算した値を 2 番目と 3 番目のバイトに移動します。3 を返して、変換された文字の長さが 3 バイトであることを示します。
  - ⑪ ワイド文字の値が 0x5f1a から 0x8df7 の範囲にあれば、計算した値を配列 *s* の最初のバイトと 2 番目のバイトに移動します。2 を返して、変換された文字の長さが 2 バイトであることを示します。
  - ⑫ `errno` に `[EILSEQ]` を設定し、-1 を返して、ワイド文字が無効であることを示します。
- これらの文は、ワイド文字の値が上記の条件をいずれも満たさない場合に実行されます。

#### 6.3.1.10 `wcswidth()` 関数用のメソッドの作成

`wcswidth()` 関数は `wcswidth` メソッドを使用して、ワイド文字列を表示するのに必要なカラム数を判定します。一般に、このメソッドの C ソース・ファイルの名前は、`__wcswidth_codeset.c` です。`codeset` は、このメソッドに対応するコードセットを示します。例 6-19 は、`ja_JP.sdeckanji` ロケールに使用される `wcswidth` メソッドを定義するファイル、`__wcswidth_sdeckanji.c` です。

例 6-19: `ja_JP.sdeckanji` ロケールのための `__wcswidth_sdeckanji` メソッド

```
#include <stdlib.h> ①
#include <wchar.h>
#include <sys/localedef.h>

/*
The algorithm for this conversion is:

PC <= 0x009f:          s[0] = PC
PC >= 0x0100 and PC <=0x015d: s[0] = 0x8e
                        s[1] = PC - 0x005f
PC >= 0x015e and PC <=0x303b: s[0] = ((PC - 0x015e) >> 7) + 0x00a1
                        s[1] = ((PC - 0x015e) & 0x007f) + 0x00a1
PC >= 0x303c and PC <=0x5f19: s[0] = 0x8f
                        s[1] = ((PC - 0x303c) >> 7) + 0x00a1
                        s[2] = ((PC - 0x303c) & 0x007f) + 0x00a1
PC >= 0x5f1a and PC <=0x8df7 s[0] = ((PC - 0x5f1a) >> 7) + 0x00a1
                        s[1] = ((PC - 0x5f1a) & 0x007f) + 0x0021

+-----+-----+-----+-----+

```

## 例 6-19: ja\_JP.sdeckanji ロケールのための \_\_wcswidth\_sdeckanji メソッド (続き)

| process code    | s[0]      | s[1]      | s[2]      |              |
|-----------------|-----------|-----------|-----------|--------------|
| 0x0000 - 0x009f | 0x00-0x9f | --        | --        |              |
| 0x00a0 - 0x00ff | --        | --        | --        |              |
| 0x0100 - 0x015d | 0x8e      | 0xa1-0xfe | --        | JIS X0201 RH |
| 0x015e - 0x303b | 0xa1-0xfe | 0xa1-0xfe | --        | JIS X0208    |
| 0x303c - 0x5f19 | 0x8f      | 0xa1-0xfe | 0xa1-0xfe | JIS X0212    |
| 0x5f1a - 0x8df7 | 0xa1-0xfe | 0x21-0xfe | --        | UDC          |

```

*/ 2

int __wcswidth_sdeckanji(
    const wchar_t *wcs, 3
    size_t n, 4
    _LC_ucharmap_t *hdl ) 5
{
    int len; 6
    int i; 7

    if (wcs == (wchar_t *)NULL || *wcs == (wchar_t)NULL)
        return(0); 8

    len = 0; 9
    for (i=0; wcs[i] != (wchar_t)NULL && i<n; i++) { 10

        if (wcs[i] <= 0x9f)
            len += 1; 11

        else if ((wcs[i] >= 0x0100) && (wcs[i] <= 0x015d))
            len += 1; 12

        else if ((wcs[i] >=0x015e) && (wcs[i] <= 0x303b))
            len += 2; 13

        else if ((wcs[i] >=0x303c) && (wcs[i] <= 0x5f19))
            len += 2; 14

        else if ((wcs[i] >=0x5f1a) && (wcs[i] <= 0x8df7))
            len += 2; 15

        else
            return(-1); 16
    } 17

    return(len); 18
}

```

- 1 このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- 2 必要な表示幅を判定するためのアルゴリズムを記述します。

各文字の表示幅は、文字が属する文字セットに応じて 1 カラムまたは 2 カラムのいずれかです。表示幅は、マルチバイト形式の文字のサイズとは異なります。たとえば、3 バイト文字には 2 つの表示カラムが必要であり、2 バイト文字には 1 つまたは 2 つの表示カラムが必要です。

- ③ 表示幅情報を必要とするワイド文字列を含むバッファを、`wcs` でポイントします。
- ④ `wcs` バッファの最大サイズを含む変数 `n` を定義します。
- ⑤ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`hdl` でポイントします。
- ⑥ バイトまたはカラム数で表示幅を保持する変数 `len` を定義します。
- ⑦ ループ・カウンタとして使用する変数 `i` を定義します。
- ⑧ `wcs` が `NUL` を含んでいるか、`NULL` をポイントしている場合は、ゼロ (0) を返します。
- ⑨ `len` をゼロ (0) に初期化します。
- ⑩ `wcs` バッファ内の各ワイド文字を処理し、ワイド文字ポインタをインクリメントする `for` ループを開始します。
- ⑪ 現在のワイド文字の値が `0x9f` 以下であれば、`len` を 1 だけインクリメントします。
- ⑫ 現在のワイド文字の値が `0x0100` から `0x015d` の範囲にあれば、`len` を 1 だけインクリメントします。
- ⑬ 現在のワイド文字の値が `0x015e` から `0x303b` の範囲にあれば、`len` を 2 だけインクリメントします。
- ⑭ 現在のワイド文字の値が `0x303c` から `0x5f19` の範囲にあれば、`len` を 2 だけインクリメントします。
- ⑮ 現在のワイド文字の値が `0x5f1a` から `0x8df7` の範囲にあれば、`len` を 2 だけインクリメントします。
- ⑯ -1 を返して、文字列に無効なワイド文字が含まれていることを示します。

この文は、文字列中の値が上記の条件をいずれも満たさない場合に実行されます。呼び出し元の関数 `wcswidth()` は、ワイド文字が印字不可

能な場合にも -1 を返します。ただし、この条件は呼び出し元関数のレベルで評価されるため、このメソッドが評価する必要はありません。

**17** wcs バッファ内のワイド文字を処理する for ループを終了します。

**18** ワイド文字列を表示するのに必要なカラム数を示す len を返します。

### 6.3.1.11 wwidth() 関数用のメソッドの作成

wwidth() 関数は wwidth メソッドを使用して、ワイド文字を表示するのに必要なカラム数を判定します。一般に、このメソッドの C ソース・ファイルの名前は、\_\_wwidth\_codeset.c です。codeset は、このメソッドに対応するコードセットを示します。例 6-20 は、ja\_JP.sdeckanji ロケールで使用される wwidth メソッドを定義するファイル、\_\_wwidth\_sdeckanji.c です。

#### 例 6-20: ja\_JP.sdeckanji ロケールのための \_\_wwidth\_sdeckanji メソッド

```
#include <stdlib.h> 1
#include <wchar.h>
#include <sys/localedef.h>

/*
The algorithm for this conversion is:

PC <= 0x009f:          s[0] = PC
PC >= 0x0100 and PC <=0x015d: s[0] = 0x8e
                          s[1] = PC - 0x005f
PC >= 0x015e and PC <=0x303b: s[0] = ((PC - 0x015e) >> 7) + 0x00a1
                          s[1] = ((PC - 0x015e) & 0x007f) + 0x00a1
PC >= 0x303c and PC <=0x5f19: s[0] = 0x8f
                          s[1] = ((PC - 0x303c) >> 7) + 0x00a1
                          s[2] = ((PC - 0x303c) & 0x007f) + 0x00a1
PC >= 0x5f1a and PC <=0x8df7 s[0] = ((PC - 0x5f1a) >> 7) + 0x00a1
                          s[1] = ((PC - 0x5f1a) & 0x007f) + 0x0021

+-----+-----+-----+-----+
| process code | s[0] | s[1] | s[2] |
+-----+-----+-----+-----+
| 0x0000 - 0x009f | 0x00-0x9f | -- | -- |
| 0x00a0 - 0x00ff | -- | -- | -- |
| 0x0100 - 0x015d | 0x8e | 0xa1-0xfe | -- | JIS X0201 RH
| 0x015e - 0x303b | 0xa1-0xfe | 0xa1-0xfe | -- | JIS X0208
| 0x303c - 0x5f19 | 0x8f | 0xa1-0xfe | 0xa1-0xfe | JIS X0212
| 0x5f1a - 0x8df7 | 0xa1-0xfe | 0x21-0xfe | -- | UDC
+-----+-----+-----+-----+

*/ 2

int __wwidth_sdeckanji(
    wint_t wc, 3
    _LC_charmap_t *hdl ) 4
{
    if (wc == 0)
        return(0); 5
```

### 例 6-20: ja\_JP.sdeckanji ロケールのための `__wcwidth_sdeckanji` メソッド (続き)

```
if (wc <= 0x9f)
    return(1); ❹

else if ((wc >= 0x0100) && (wc <= 0x015d))
    return(1); ❺

else if ((wc >=0x015e) && (wc <= 0x303b))
    return(2); ❽

else if ((wc >=0x303c) && (wc <= 0x5f19))
    return(2); ❾

else if ((wc >=0x5f1a) && (wc <= 0x8df7))
    return(2); ❿

return(-1); ⓫
}
```

- ❶ このメソッドに必要な定数と構造体を含むヘッダ・ファイルをインクルードします。
- ❷ 必要な表示幅を判定するためのアルゴリズムを記述します。  
各文字の表示幅は、文字が属する文字セットに応じて1カラムと2カラムのいずれかです。表示幅は、マルチバイト形式の文字のサイズとは異なります。たとえば、3バイト文字には2つの表示カラムが必要であり、2バイト文字には1つまたは2つの表示カラムが必要です。
- ❸ 表示幅情報を必要とするワイド文字を含む `wc` 変数を定義します。
- ❹ このロケールの文字マップを解析するメソッドへのポインタを格納している構造体を、`hdl` でポイントします。
- ❺ ワイド文字バッファが空の場合、ゼロ (0) を返します。
- ❻ ワイド文字値が `0x009f` 以下の場合、1 を返します。
- ❼ ワイド文字値が `0x0100` から `0x015d` の範囲にあれば、1 を返します。
- ❽ ワイド文字値が `0x015e` から `0x303b` の範囲にあれば、2 を返します。
- ❾ ワイド文字値が `0x303c` から `0x5f19` の範囲にあれば、2 を返します。
- ❿ ワイド文字値が `0x5f1a` から `0x8df7` の範囲にあれば、2 を返します。
- ⓫ ワイド文字の値が無効な場合、-1 を返します。

呼び出し元の関数 `wcwidth()` は、ワイド文字が印字不可能な場合にも `-1` を返します。ただし、この条件は呼び出し元関数のレベルで評価されるため、このメソッドが評価する必要はありません。

### 6.3.2 オプションのメソッド

ロケールには、必須メソッド (6.3.1 項を参照) 以外のメソッドも含めることができます。メソッド指定がないために省略時のメソッドが適用された場合、そのメソッドはオプションだと見なされます。つまり、ロケールでメソッドを使用するが、特定のロケール・カテゴリに関連する関数や、その他のロケール関連の関数用のメソッドを用意していない場合、`localedef` コマンドは、シングルバイト文字とマルチバイト文字の両方の処理コードを扱う省略時のメソッドを適用します。

オプションのメソッドを作成するためには、C ライブラリ・ルーチンの内部インタフェースについての詳細情報が必要になります。この情報はベンダ独自の情報であり、変更される場合があります。したがって、この項のオプション・メソッドの説明は、必須メソッドの説明と比べて不完全です。

まれなケースとして、オプションのメソッドをロケールに含めなければならないことがあります。詳細は、最寄りの技術サポートにお問い合わせください。

以下に、オプションのメソッドを示します。

- `LC_CTYPE` カテゴリ
  - `towupper`
  - `towlower`
  - `wctype`
  - `iswctype`
- `LC_COLLATE` カテゴリ
  - `fnmatch`
  - `strcoll`
  - `strxfrm`
  - `wscoll`
  - `wcsxfrm`

- regcomp
- regexec
- regfree
- regerror
- LC\_MONETARY カテゴリ, LC\_NUMERIC カテゴリ, または両方
  - localeconv
  - strfmon
- LC\_TIME カテゴリ
  - strftime
  - strptime
  - wcsftime
- LC\_MESSAGES カテゴリ
  - rpmatch
- その他
  - nl\_langinfo()

### 6.3.3 ロケールで使用するシェアード・ライブラリの作成

例 6-21 に, コンパイラとリンカのコマンド行を示します。これらのコマンド行は, `ja_JP.sdeckanji` ロケールで使用するシェアード・ライブラリをメソッドのソース・ファイルから作成する際に必要です。

**例 6-21: `ja_JP.sdeckanji` ロケールで使用するメソッドのライブラリの作成**

---

```
cc -std0 -c \
__mblen_sdeckanji.c __mbstopcs_sdeckanji.c \
__mbstowcs_sdeckanji.c __mbtopc_sdeckanji.c \
__mbtowc_sdeckanji.c __pcstombs_sdeckanji.c \
__pctomb_sdeckanji.c __wcstombs_sdeckanji.c \
__wcswidth_sdeckanji.c __wctomb_sdeckanji.c \
__wcwidth_sdeckanji.c

ld -shared -set_version osf.1 -soname libsdeckanji.so -shared \
-no_archive -o libsdeckanji.so \
__mblen_sdeckanji.o __mbstopcs_sdeckanji.o \
__mbstowcs_sdeckanji.o __mbtopc_sdeckanji.o \
__mbtowc_sdeckanji.o __pcstombs_sdeckanji.o __pctomb_sdeckanji.o \
__wcstombs_sdeckanji.o __wcswidth_sdeckanji.o __wctomb_sdeckanji.o \
__wcwidth_sdeckanji.o \
```



### 例 6-21: ja\_JP.sdeckanji ロケールで使用されるメソッドのライブラリの作成 (続き)

```
-lc
```

シェアード・ライブラリについての詳細は、cc(1) および ld(1) のリファレンス・ページを参照してください。

## 6.3.4 ロケールに対応した methods ファイルの作成

methods ファイルには、ロケールで使用されるメソッド・シェアード・ライブラリで定義されている各関数のエントリが含まれています。関数が実行する操作は、メソッド・キーワードで識別されます。メソッド・キーワードの後には、引用符で囲まれた関数の名前と、その関数を含むシェアード・ライブラリのパスが続きます。

例 6-22 に、ja\_JP.sdeckanji ロケールで使用されるメソッドのための methods ファイルのセクションを示します。C ライブラリ・インタフェースの指定を変更したいときは、必須のメソッドのリストを定義する必要があるため、この例で示すように、methods ファイルには個々の必須メソッドのエントリが含まれていなければなりません。ja\_JP.sdeckanji ロケールではすべてのオプション・メソッドに対して、省略時の実装が使用されています。そのため、この例にはオプション・メソッドのエントリは含まれていません。

### 例 6-22: ja\_JP.sdeckanji ロケールのための methods ファイル

```
# sdeckanji.m 1
# <method_keyword> "<entry>" "<package>" "<library_path>" 1

METHODS 2

__mbstopcs "__mbstopcs_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
__mbtopc "__mbtopc_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
__pcstombs "__pcstombs_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
__pctomb "__pctomb_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
mblen "__mblen_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
mbstowcs "__mbstowcs_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
mbtowc "__mbtowc_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" 3
wcstombs "__wcstombs_sdeckanji" "libsdeckanji.so" \
```

### 例 6-22: ja\_JP.sdeckanji ロケールのための methods ファイル (続き)

```
"/usr/shlib/libsdeckanji.so" ③
wcswidth  "__wcswidth_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" ③
wctomb    "__wctomb_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" ③
wcwidth   "__wcwidth_sdeckanji" "libsdeckanji.so" \
"/usr/shlib/libsdeckanji.so" ③

END METHODS ④
```

#### ① コメント行

これらの行は、methods ファイルの名前と、メソッド・エントリのフォーマットを示します。フォーマット中の <package> フィールドは無視されますが、ライブラリ・パスを指定するためには、何らかの文字列をこのフィールドに設定しておかなければなりません。

#### ② メソッド・エントリの開始を示すヘッダ

#### ③ 必須メソッドのエントリ

#### ④ メソッド・エントリの終わりを示すトレイラ

methods ファイルのエントリの詳細については、`localedef(1)` のリファレンス・ページを参照してください。

## 6.4 ロケールの作成とテスト

`localedef` コマンドを使用して、ソース・ファイルからロケールを作成します。例 6-23 は、この章のほとんどの例で使用されているフランス語ロケールを作成するためのコマンド行です。この例では、すべてのソース・ファイルがユーザの省略時のディレクトリに置かれており、作成されたロケールもそのディレクトリに置かれるものとします。

### 例 6-23: fr\_FR.ISO8859-1@example ロケールの作成

```
% localedef -f ISO8859-1.cmap \ ①
-i fr_FR.ISO8859-1.src \ ②
fr_FR.ISO8859-1@example ③
```

- ❶ `-f` オプションは、文字マップ・ソース・ファイルを指定します。
- ❷ `-i` オプションは、ロケール定義ソース・ファイルを指定します。
- ❸ コマンドに指定する最後の引数は、ロケールの名前です。

ロケールをテストする場合、特に、そのロケールがシステム上にインストールされている標準ロケールに類似している場合は、そのロケール名に拡張子を付けます。アットマーク (@) 拡張子を付加した名前を使用することにより、言語、地域、およびコードセットに対して標準の文字列を指定でき、またテスト・ロケールを一意に識別できます。この設定は、後でテスト・ロケールを、他のロケールが配置されているディレクトリ `/usr/lib/nls/loc` に移動する際に重要になります。

例 6-23 には、`localedef` コマンドの 1 つの形式と、2, 3 のオプションだけを示します。`localedef(1)` のリファレンス・ページにこのコマンドの詳細な説明があります。

以下に、重要な規則とオプションの要約を示します。

- ロケールでメソッドが定義されている場合は、`-m` オプションを付けて `methods` ファイルを指定しなければならない。たとえば、`ja_JP.sdeckanji` ロケールを作成するためのコマンド行には、例 6-22 に示すファイルを使用するために `-m sdeckanji.m` が指定されています。
- デバッグを行うときは、`-v` オプションを指定することにより、コマンドを詳細モードで実行できる。このオプションを `-c` オプションとともに使用すると、ロケールに関して有用な情報を含む `.c` ファイルが作成されます。
- 重複する定義が見つかったときに警告を表示するには、`-w` オプションを指定する。

省略時の設定では、ロケールは `/usr/lib/nls/loc` ディレクトリになければなりません。テスト・ロケールを `/usr/lib/nls/loc` ディレクトリに移動する前にテストしたい場合は、`LOCPATH` 変数を設定し、ロケールが置かれているディレクトリを指定します。その後で `LANG` 環境変数を新しいロケールに設定すれば、コマンドやアプリケーションを使用してロケールを対話形式でテストできます。

例 6-24 は、date コマンドを使用して、日付と時刻の形式をテストします。

#### 例 6-24: LOCPATH 変数の設定とロケールのテスト

---

```
% setenv LOCPATH ~harry/locales
% setenv LANG fr_FR.ISO8859-1@example
% date
ven 23 avr 13:43:05 EDT 1999
```

---

#### 注意

LOCPATH 変数は、X/Open の UNIX 標準で規定されている仕様の拡張であり、この仕様に準拠するすべてのシステムで認識されるわけではありません。

プログラムによっては、標準ロケールの名前とまったく同じ名前を持つシステム・ディレクトリにインストールされたサポート・ファイルが使用されています。その場合、アプリケーション・ソフトウェアやシステム・ソフトウェア、あるいはその両方は、LANG 環境変数の値を使用して、サポート・ファイルが置かれているロケール固有のディレクトリを決定します。LANG または LC\_ALL 環境変数にロケール・ファイル名を直接指定した場合、アットマーク (@) サフィックスが付加されているロケール・ファイル名は、アプリケーションによっては無効な検索パスとなることがあります。

LANG 環境変数に標準のロケール名を指定し、ロケールのカテゴリ変数に異なるロケール名を指定することにより、この問題を回避する方法を次の例に示します。指定しなければならないカテゴリ変数は、その適用範囲が作成したロケールと元のロケール間で異なっているものだけです。

```
% setenv LANG fr_FR.ISO8859-1
% setenv LC_CTYPE fr_FR.ISO8859-1@example
% setenv LC_COLLATE fr_FR.ISO8859-1@example
:
% setenv LC_TIME fr_FR.ISO8859-1@example
```

## 国際化アプリケーションのプログラミング上の注意

この章では、国際化アプリケーションを開発する上で考慮しなければならないさまざまな作業について説明します。次のような作業があります。

- 入力システムと入力スタイルの選択 (7.1 節)
- ユーザ定義文字データベースの管理 (7.2 節)
- ロケールの指定によるソート順序の割り当て (7.3 節)
- 英語以外の言語におけるリファレンス・ページの処理 (7.4 節)
- データ・ファイルのコードセットの変換 (7.5 節)
- 中国語および韓国語の PostScript サポートでのフォント・レンダラの使用 (7.6 節)

この章では、国際化アプリケーションの作成に必要なツールに関する情報を説明します。この章で説明する内容は、国際化アプリケーションをオペレーティング・システム上で使用する方法にも深く関係します。この章の情報を使用する時、姉妹編の『国際化機能ユーザズ・ガイド』も参照すると役に立ちます。

次のマニュアルでは、オペレーティング・システム上でアジア系言語用に提供されているカスタマイズ方法とソフトウェアの使用方法について、言語ごとに説明しています。

- 『*Technical Reference for Using Chinese Features*』
- 『日本語機能ガイドブック』
- 『*Technical Reference for Using Korean Features*』
- 『*Technical Reference for Using Thai Features*』

これらのマニュアルは、本オペレーティング・システムのドキュメント Web サイト ([http://www.tru64unix.com-paq.com/docs/pub\\_page/V51A\\_DOCS/PRG\\_DOCS.HTM](http://www.tru64unix.com-paq.com/docs/pub_page/V51A_DOCS/PRG_DOCS.HTM)) で、プログラミング

グ関連ドキュメントとして掲載されています。中国語、日本語、および韓国語のテクニカル・リファレンスのテキストでは、英語以外の文字が使用されています。これらの文字を Web ブラウザで表示するには、適切な言語サポート・サブセットをシステム上にインストールする必要があります。また、ロケールには、テクニカル・リファレンスで使用するローカル言語文字を含むロケールを設定する必要があります。

本オペレーティング・システムのドキュメントには、サポートされているすべての言語およびコードセットのリファレンス・ページの他に、国際化 (i18n\_intro(5)) や、地域化 (l10n\_intro(5)) に関する入門編のリファレンス・ページがあります。

## 7.1 入力システムの選択

日本語や中国語、韓国語などの言語では、文字や語句を入力するのに入力システムを使用します。入力システムを使用すると、入力データにさまざまな編集を加えて文字の入力が行えます。文字入力の途中で入力されるデータは、前編集文字列と呼ばれます。

X ウィンドウの入力システム仕様では、次のユーザ入力 (前編集) スタイルが規定されています。

- On-the-Spot スタイル

編集中のデータは、アプリケーション・ウィンドウに直接表示されます。アプリケーション・データは、前編集文字列が文字の入力位置に表示されるように移動されます。

- Over-the-Spot スタイル

前編集文字列は、入力位置の上に重なったウィンドウ内に表示されます。

- Off-the-Spot スタイル

前編集文字列は、アプリケーション・ウィンドウ内の、入力位置に重ならないウィンドウ内に表示されます。一般に、前編集文字列用ウィンドウは、アプリケーション・ウィンドウの最下部に表示されます。この場合、前編集ウィンドウがアプリケーション・ウィンドウの最後のテキスト行を隠してしまうことがありますが、アプリケーション・ウィンドウをサイズ変更すれば、最後の行を表示できます。

- Root-Window スタイル

前編集文字列は、アプリケーションの外、つまりルート・ウィンドウ上の子ウィンドウに表示されます。

一般に、各ロケールにおける入力システムは、複数のユーザ入力スタイルをサポートしますが、すべての入力スタイルをサポートしているわけではありません。入力システムによってサポートされている言語で操作を行うときは、VendorShell リソースの `XmNpreeditType` を使用して、入力スタイルの優先順位を指定できます。省略時の設定では、このリソースは次のように定義されています。

`OnTheSpot, OverTheSpot, OffTheSpot, Root`

これらの値の優先順位は、入力システムがサポートしていれば `On-the-Spot` スタイルが使用され、`On-the-Spot` がサポートされず `Over-the-Spot` がサポートされていれば、`Over-the-Spot` が使用されることを示します (`Off-the-Spot` と `Root-Window` に関しても同様)。

`XmNpreeditType` リソース値をアプリケーションに渡すには、次のいずれかの方法を使用します。

- CDE アプリケーションの入力システムを使用する。このアプリケーションの使用方法については、『*CDE ガイドブック*』を参照してください。
- アプリケーション固有のリソース・ファイルで指定する。
- コマンド行でアプリケーションを実行する。

```
% app-name -xrm '*preeditType: offthespot,onthespot' &
```

入力スタイルは、専用の入力サーバによってサポートされます。入力サーバは独立したプロセスとして動作し、アプリケーションと通信しながら入力操作を処理します。

入力サーバは、アプリケーションと同じシステムで実行している必要はありませんが、1 つの例外を除き、アプリケーションを起動する前に実行されており、アプリケーションからアクセス可能でなければなりません。

Motif アプリケーションが、国際化されて簡体字中国語をサポートしており、`Reconnectable` リソースに `True` が設定されている `XmText` または `XmTextField` ウィジェットを含んでいる場合、アプリケーションは入力サーバとの接続を、アプリケーションの最初の起動時、またはサーバの停止および再起動時に確立することができます。詳細については、`XmText(3X)` および `XmTextField(3X)` のリファレンス・ページを参照してください。

本オペレーティング・システムで利用可能な入力サーバや、各サーバがサポートしている入力スタイルについては、『国際化機能ユーザズ・ガイド』を参照してください。

## 7.2 ユーザ定義文字と語句入力の管理

日本、台湾、および中国で使用される文字セットには、アジア地域の場所の名前や人名に使用されることがある一部の文字が含まれていません。そのような文字はユーザが定義でき、サイト別のデータベースに格納されます。そのようなデータベースは、ユーザ定義文字 (UDC) または文字属性データベースと呼ばれます。表意文字を定義するときは、その文字のフォント・グリフ、照合ファイル、およびその他のサポート・ファイルも定義しなければなりません。

UDC データベースの設定方法および使用方法については付録 B を参照してください。

韓国、台湾、および中国のユーザは、キーワード、省略形、あるいは頭字語を使用して、完全な語句を入力できます。この機能は、語句データベースと入力メカニズムによってサポートされます。語句データベースの設定および使用方法については、『国際化機能ユーザズ・ガイド』を参照してください。

/var/i18n/conf/cp\_dirs 構成ファイルによって、ソフトウェア・サービスやハードウェアに、UDC や語句入力をサポートするデータベースを検索させることができます。

例 7-1 に、cp\_dirs ファイルの省略時のエントリを示します。これらのエントリを編集し、省略時の位置を変更できます。

例 7-1: 省略時の cp\_dirs ファイル

---

```
#
# Attribute directory configuration file
#
#           System location           User location
#           =====
udc    -           /var/i18n/udc           ~/.udc
odl    -           /var/i18n/odl           ~/.odl
sim    -           /var/i18n/sim           ~/.sim
cdb    /usr/i18n/.cdb /var/i18n/cdb           ~/.cdb
iks    -           /var/i18n/iks           ~/.iks
pre    -           /var/i18n/fonts         ~/.fonts
bdf    -           /var/i18n/fonts         ~/.fonts
pcf    -           /var/i18n/fonts         ~/.fonts
```

---



cp\_dirs ファイル内の個々の行は、1つのエントリを表しており、次のフォーマットを持ちます。

[*service\_name standard\_path system\_path user\_path*]

*service\_name* は次のいずれかです。

- bdf (bdf フォーマットのフォント・ファイル)
- cdb (asort コマンドで使用する照合値データベース)
- iks (入力キー・シーケンス・ファイル)
- odl (SoftODL サービスが使用するフォントと入力キー・シーケンスのデータベース)
- pcf (Printer Customization File フォーマットのフォント・ファイル)  
これらのファイルは、フォント解像度に応じて、サブディレクトリの 75dpi または 100dpi に置かれます。
- pre (cgen コマンドで作成されたプリロード・フォーマットのフォント・ファイル)  
これらのファイルは、マルチバイト文字端末にプリロードされる raw フォント・ファイルです。
- sim (語句データベース)
- udc (UDC データベース)

cp\_dirs ファイルには、各サービスについて1つのエントリしか含められません。エントリ行の残りのフィールドは、次のパス指定で構成されます。

- *standard\_path* は、標準文字セット用の照合値データベースの位置を指定する (cdb エントリにのみ適用される)。
- *system\_path* は、システム全体のデータベースの位置を指定する。
- *user\_path* は、ユーザの個人用データベースの位置を指定する。

上記の位置は、次のいずれかの形式で指定します。

- 絶対パス名 (/) で始まるパス名
- ユーザのホーム・ディレクトリを基準とするパス名 (~ / で始まるパス名)
- エントリが使用されていないことを示す - (マイナス記号またはハイフン)

たとえば、ユーザ定義文字がシステム全体のデータベースでのみサポートされるようにしたいときは、ユーザ定義文字に関連するすべてのサービスについて `user_path` に `-` を指定します。

`cp_dirs` ファイル内のコメント行は、番号記号 (`#`) で始まります。

## 7.3 ロケールの指定によるソート順序の割り当て

`sort` コマンドは、現在のロケールで定義されている照合順序に従って文字をソートします。ロケールによっては、対応する文字セットに対して複数の照合規則を適用できます。ただし、その場合には、言語、地域、およびコードセットの同じ組み合わせについて、複数のロケール名が存在します。これらのバリエーションは、ユーザが複数の照合順序を選択できるようにするためのものです。

言語、地域、およびコードセットの組み合わせについて複数のロケールが存在する場合、`@variant` 形式のサフィックスが付加されることがあります。一般に、`%L` 指定子を使用して作成されるパス名で問題が生じないようにするために、`@` サフィックスが付加されているロケール名は、適切なロケール・カテゴリ変数にのみ指定するようにします。次の例では、`LC_COLLATE` に割り当てられるロケールと `LANG` に割り当てられるロケールは、照合順序のみが異なります。

```
% setenv LANG zh_TW.eucTW
% setenv LC_COLLATE zh_TW.eucTW@radical
```

ほとんどの言語では、1 つまたは複数のロケールを介して、複数の照合順序をサポートするのが適切な方法です。ただし、アジア系言語における照合順序には、次に示す理由から、さらに追加のサポートが必要になります。

- アジア系言語には、ロケールで規定されていない UDC が含まれている。これらの文字は照合重み付きで定義できます。その場合、ソート対象の文字列にユーザ定義文字が現れたときは、その重みを適用しなければなりません。
- 表意文字は、部首、画数、発音、内部コードなど、異なる要素に従ってソートできる。ソート操作の際に、これらの要素を組み合わせる必要が生じる場合があります。操作によっては、まず文字を部首でソートし、次に画数でソートしなければならないことがあります。また別の操作では、まず文字を発音順にソートして、次に画数の順に並べなくてはならないこともあります。要素を組み合わせるソートには、深さ

優先のソートではなく、幅優先のソートがロケールでサポートされていなければなりません。

これらの理由から `asort` コマンドが提供されており、アジア系言語をサポートする言語別サブセットをインストールすれば、このコマンドが使用できるようになります。`asort` コマンドは、省略時の設定では、`LC_COLLATE` 変数で定義されている照合順序を使用し、`sort` コマンドがサポートするすべてのフラグをサポートします。また、`asort` コマンドには次のフラグが追加されています。

- `-C`

UDC をサポートするために、ソート操作が、`cgen` ユーティリティによって生成されたソート・テーブルに加え、特殊なシステム・ソート・テーブルを使用することを示します。このフラグは、`LC_COLLATE` 変数によって指定されたロケールで定義されているソート・シーケンスを無効にします。

- `-v`

`-C` フラグとともに指定することにより、幅優先のソートを可能にします。

このコマンドの使用方法については、`asort(1)` のリファレンス・ページを参照してください。

## 7.4 英語以外の言語におけるリファレンス・ページの処理

一般に、UNIX システムのアプリケーションを開発するプログラマは、アプリケーションとそのコンポーネントを解説するためのオンラインのリファレンス・ページ (manpage) を作成します。国際市場でアプリケーションを販売するためには、UNIX のテキスト処理用のコマンドとユーティリティで、翻訳版のリファレンス・ページが扱えなければなりません。このニーズに応えるために、Tru64 UNIX には `nroff`、`tbl`、および `man` コマンドの拡張版が用意されています。

### 7.4.1 `nroff` コマンド

`nroff` コマンドは、ロケールをサポートするための次の機能を備えています。

- システムにインストールされているロケールに対応する言語で書かれたリファレンス・ページのソース・ファイルをフォーマットする。

- マクロと基本コマンドの文字列引数中で、サポートされているすべての言語の文字を処理する。
- リファレンス・ページのソース・ファイル中の `.tr` コマンドにより、サポートされているすべての言語の文字の文字マッピングをサポートする。
- ASCII 文字に加え、ローカル言語のエスケープ文字 (`\`)、コマンド制御文字 (`.`)、および `nobreak` 制御文字 (`'`) の設定を可能にする。
- アジア系言語のほとんどのコードセットで定義されている 2 バイトのスペース文字を、出力中で 2 つの ASCII スペースにマップする。

表意文字を含むリファレンス・ページをフォーマットする場合、`nroff` コマンドは各文字を 1 つの単語として扱います。2 バイトの英字と句読点を含む表意文字の文字列は、以下の制約に基づいて次の行に折り返されます。

- テキスト行の最後の文字は、標準または個人用の行末禁則文字リストのいずれにおいても、行末禁則文字として定義される文字であってはならない。
- テキスト行の先頭の文字は、標準または個人用の行頭禁則文字リストのいずれにおいても、行頭禁則文字として定義される文字であってはならない。

標準の行頭禁則文字と行末禁則文字のリストは、`nroff` カタログ・ファイルで定義されています。これらの文字の詳細については、次の言語固有のマニュアルを参照してください。

- 『*Technical Reference for Using Chinese Features*』
- 『日本語機能ガイドブック』
- 『*Technical Reference for Using Korean Features*』
- 『*Technical Reference for Using Thai Features*』

これらのマニュアルは、本オペレーティング・システムのドキュメント Web サイト ([http://www.tru64unix.com-paq.com/docs/pub\\_page/V51A\\_DOCS/PRG\\_DOCS.HTM](http://www.tru64unix.com-paq.com/docs/pub_page/V51A_DOCS/PRG_DOCS.HTM)) で、プログラミング関連ドキュメントとして掲載されています。

行頭禁則文字と行末禁則文字は、`nroff` が句読点や右カッコをテキスト行の先頭に置いたり、左カッコをテキスト行の末尾に置くことを防ぐためのものです。標準の制約は、ソース・ファイル中に `.ki` と `.ko` コマンドを埋め込むことにより、オン/オフできます。

また、次のコマンドを使用して、行頭禁則文字と行末禁則文字の個人用セットを定義することもできます。

```
.kl 'no-first-list'no-last-list '
```

パラメータ *no-first-list* と *no-last-list* は、行頭禁則文字と行末禁則文字のカテゴリに含める文字の並びです。個人用の行頭禁則文字と行末禁則文字のリストを取り消すには、パラメータにヌル文字列を指定して *.kl* コマンドを入力します。次に例を示します。

```
.kl ''
```

---

#### 注意

---

*.kl* コマンドで指定した文字は、行頭禁則文字と行末禁則文字の標準セットを補足するのではなく、上書きします。そのため、行頭禁則文字と行末禁則文字の標準セットを個人用セットと併用することはできません。

コマンド *.kl ''* を入力すると、現在のロケールにおける標準の行頭禁則文字と行末禁則文字セットが使用されます。

---

*nroff* コマンドでは、テキストを右揃えにすることも、右揃えにしないことも可能です。テキストを右揃えにすると、*nroff* は行中の単語の間にスペースを挿入します。表意文字は、フォーマット処理のほとんどの過程で単語として扱われますが、スペースで区切ることが可能かどうかという点で、文字ごとに違いがあります。前にスペースを置ける文字、後にスペースを置ける文字、あるいはこの両方が可能な文字のリストが、言語別ユーザ・マニュアルに記載されています。このユーザ・マニュアルは、Tru64 UNIX の言語別サブセットをインストールすると、オンラインで利用できます。テキストを右揃えにする場合、*nroff* コマンドは次の位置にだけスペースを挿入します。

- 1 バイトまたは 2 バイトのスペースがすでに存在する位置
- 英語の文字と表意文字の間
- 前にスペースを置くことが可能な文字として定義されている文字の前
- 後にスペースを置くことが可能な文字として定義されている文字の後

これ以外の場合、連続した表意文字の間にはスペースは挿入されません。したがって、テキスト行が表意文字だけを含んでいるときは、右揃えが不可能な場合があります。

### 7.4.2 tbl コマンド

tbl コマンドは、.TS と .TE マクロで区切られたブロック内で、表フォーマット・コマンドの前処理を実行します。tbl コマンドは、英語以外の言語のテキストに現れるマルチバイト文字を扱えます。

tbl コマンドは、neqn (数式フォーマット・プリプロセッサ) と一緒に使用されることが多く、nroff コマンドに渡された入力に対してフィルタ処理を行います。その場合、最初に tbl を指定して、パイプを通して渡されるデータ量を最小限に抑えます。次に例を示します。

```
% cd /usr/share/ja_JP.deckanji/man/man1
% tbl od.1 | neqn | nroff -Tlpr -man -h | lpr -Pmyprinter
```

アジア系言語のテキストをプリントするときは、その言語をサポートしているプリンタを使用しなければなりません。

### 7.4.3 man コマンド

man コマンドは、リファレンス・ページ・ファイル内のマルチバイト文字を扱えます。省略時の設定では、man コマンドは、/usr/share/man と /usr/local/man ディレクトリを検索する前に、/usr/share/locale\_name/man ディレクトリにあるリファレンス・ページを自動的に検索します。したがって、インストールされているロケールが LANG 環境変数に設定されており、そのロケールに対応するリファレンス・ページの翻訳版が存在する場合、man コマンドは適切な言語でリファレンス・ページを表示します。

さらに、特定の言語用のリファレンス・ページの翻訳版が、ユーザのロケールのコードセットと合わないコードセットでエンコードされているときは、man コマンドは自動的にコードセット変換を適用します (適切なコンバータが利用可能な場合)。man コマンドの検索パスの再設定や、コードセット変換の詳細については、man(1) のリファレンス・ページを参照してください。

## 7.5 データ・ファイルのコードセットの変換

各ロケールは、特定のコードセットに基づいています。そのため、あるコードセットでエンコードされているデータを含むファイルを使用するアプリケーションを、別のコードセットに基づくロケールで実行する場合には、文字の解釈が正しく行えないことがあります。たとえば、ある仮想的な言語に“quo”という文字があり、その文字が、あるコードセットでは\031、別のコードセットでは\042としてエンコードされているとします。文字“quo”がデータ・ファイル内で\031として格納されている場合、そのファイルからデータを読み込むアプリケーションは、同じコードセットに基づくロケールで実行されなければなりません。さもないと、\031は“quo”以外の文字として認識されます。

ユーザ、あるいはユーザが実行するアプリケーションは、プロセス環境を特定のロケールに設定して、ロケールに基づくコードセットとは異なるコードセットで作成されたデータ・ファイルを使用しなければならない場合があります。異なる言語環境や、ユーザのロケールとは異なるコードセットでデータ・ファイルが使用される例を、次に示します。

- データ・ファイルが、ベンダ固有のコードセットに基づくロケールを使用して、ベンダのシステム上で作成されている場合。たとえば、PCをエンタープライズ・コンピューティング環境に統合する際に、UNIXユーザが、MS-DOSコードページ・フォーマットでエンコードされているデータ・ファイルを扱わなければならないことが多々あります。
- ロケールが、日本語などの同じアジア系言語をサポートする複数のUNIXロケールの1つである場合。一般にアジア系言語は、それぞれ異なるコードセットに基づく複数のロケールでサポートされています。
- データ・ファイルが、Unicode、UCS-4、UTF-8、UTF-16、UTF-32のいずれかのフォーマットでエンコードされている場合。データ・ファイル内の文字をプリントしたり、画面上に表示するのであれば、利用可能なフォントに対応するエンコーディングに文字を変換しなければなりません。

`iconv` コマンドや、`iconv_open()`、`iconv()`、および `iconv_close()` 関数を使用することにより、データ・ファイルのあるコードセットから別のコードセットに変換できます。たとえば次のコマンドは、SJISコードセットでエンコードされているファイル `accounts_local` 内のデータを

読み込み、そのデータを eucJP コードセットに変換して、結果をファイル accounts\_central に追加します。

```
% iconv -f SJIS -t eucJP accounts_local \  
>> accounts_central
```

man コマンドや国際化対応のプリント・フィルタなど、多数のコマンドおよびユーティリティで、iconv() 関数と、関連するコンバータを使用して、コードセットの変換を行います。

iconv コマンドおよび関連する関数は、データを変換する際にアルゴリズムに基づくコンバータか、テーブル・コンバータのいずれかを使用できます。アルゴリズムに基づくコンバータは、システムにインストールされていれば、/usr/lib/nls/loc/iconv ディレクトリに置かれています。このディレクトリは、コンバータを探すときに最初に検索されるディレクトリです。このディレクトリには、同じコンバータに対して、システムで指定された別の名前をマップするための別名ファイル (iconv.alias) も含まれています。テーブル・コンバータは、システムにインストールされていれば、/usr/lib/nls/loc/iconvTable ディレクトリに置かれています。LOCPATH 変数が定義されている場合、この変数の値はコマンドの省略時の検索パスに置き換わりします。

iconv コマンドは、コンバータ名が次のフォーマットになっているものと想定します。

*from-codeset\_to-codeset*

上記の例では、iconv コマンドは コンバータ /usr/lib/nls/loc/iconv/SJIS\_eucJP を検索して使用します。

また、HKSCS (Hong Kong Supplementary Character Set) のコードセット変換をサポートするオペレーティング・システムがあるとします。HKSCS は、ロケールでも文字セット名でもありませんが、中国語で行われる電子通信やデータ交換の共通言語インタフェースを提供するために使用されます。HKSCS の文字は、コンピュータでのみ使用されます。Tru64 UNIX では、HKSCS は、HKSCS 文字を含み、サポートが HKSCS と Unicode 間のコード変換に限定される、拡張 Big-5 エンコーディングの名前として使用されます。iconv コマンドを使用すると、HKSCS によるコードセット変換は、次のいずれかとして指定されます。

- UTF-16\_HKSCS または HKSCS\_UTF-16



- UCS-4\_HKSCS または HKSCS\_UCS-4
- UTF-8\_HKSCS または HKSCS\_UTF-8

Hong Kong Supplementary Character Set についての詳細は、HKSCS(5) のリファレンス・ページを参照してください。

表 7-1 に、Tru64 UNIX が英語のデータに関してサポートするコードセット変換を示します。次のマニュアルには、アジア系言語でサポートされているコードセット変換テーブルの説明があります。

- 『*Technical Reference for Using Chinese Features*』
- 『日本語機能ガイドブック』
- 『*Technical Reference for Using Korean Features*』
- 『*Technical Reference for Using Thai Features*』

iconv コマンドについての詳細は、iconv(3) および iconv\_intro(5) のリファレンス・ページを参照してください。プログラムがコードセット変換に使用できる関数については、iconv\_open(3)、iconv(1)、および iconv\_close(3) のリファレンス・ページを参照してください。特定の言語で使用可能なコードセット・コンバータについては、その言語のリファレンス・ページを参照してください。

表 7-1: 英語でサポートされているコードセット変換

| コードセット       | ASCII-GR | ISO8859-1 | ISO8859-1-GL | ISO8859-1-GR |
|--------------|----------|-----------|--------------|--------------|
| ASCII-GR     | –        |           | ×            | ×            |
| ISO8859-1    |          | –         |              |              |
| ISO8859-1-GL | ×        |           | –            | ×            |
| ISO8859-1-GR | ×        |           | ×            | –            |

## 7.6 中国語および韓国語の PostScript サポートでのフォント・レンダラの使用

この節では、中国語と韓国語の PostScript フォントをサポートする Motif アプリケーションを作成する場合の、フォント・レンダラの使用について説明します。表意文字用のキャッシュ・サイズの調整やローカル言語用のウィンドウのカスタマイズについては、『国際化機能ユーザーズ・ガイド』を参照してください。

## 7.6.1 マルチバイト PostScript フォント用のフォント・レンダラの使用法

オペレーティング・システムには、中国語と韓国語で利用可能な PostScript フォントを X アプリケーションで使えるようにするレンダラが備わっています。システム管理者は、X サーバやフォント・サーバを介して使用する、次の種類のフォントに対応するレンダラをセットアップできます。

- 2 バイト PostScript アウトライン・フォント
- UDC フォント

IOSWWXFR\*\* サブセットをインストールすると、PostScript アウトライン・フォントのフォント・レンダリングが自動的に使用できるようになります。

### 7.6.1.1 2 バイト PostScript フォント用のフォント・レンダラのセットアップ

中国語と韓国語の PostScript フォント用のフォント・レンダラは、適切な構成ファイルを編集することにより、X サーバまたはフォント・サーバを介して使用するようにセットアップできます。

- X サーバの場合、フォント・レンダラはインストール時に、X サーバの構成ファイル内の `font_renderers` リストに自動的に追加される。
- フォント・サーバの場合、フォント・サーバの構成ファイル内の `renderers` リストに、次のエントリを手作業で追加しなければならない。

```
renderers = other_renderer, other_renderer,...  
libfr_DECpscf.so;DECpscfRegisterFontFileFunctions
```

さらに、同じ構成ファイル内の `catalogue` リストに PostScript フォント・ファイルのパスを指定しなければなりません。アジア系言語用の 2 バイト PostScript フォントは、次のディレクトリに置かれています。

```
/usr/i18n/lib/X11/fonts/KoreanPS  
/usr/i18n/lib/X11/fonts/SChinesePS  
/usr/i18n/lib/X11/fonts/TChinesePS
```

これらのディレクトリにある各フォントには、次のコンポーネントが必要です。

- .pfa2 ファイル名拡張子の付いた Type1 フォント・ヘッダ  
このヘッダ・ファイルは、フォント・ディレクトリの `fonts.dir` ファイル内になければならない唯一のファイルです。
- .csdata ファイル名拡張子の付いたデータ・ファイル

- .xafm ファイル名拡張子の付いたバイナリ・メトリックス・ファイル

アジア系言語用の 2 バイト PostScript フォントのレンダラは、次の情報を含むそれ自体の構成ファイルを持ちます。

- キャッシュ・サイズ (キャッシュ・ユニットの数)
- キャッシュ・ユニットのサイズ
- ファイル・ハンドラ (フォント・レンダリング・ソフトウェアに対応する名前)
- 省略時の文字 (グリフのない文字の代わりにプリントされる文字)

この構成ファイルの省略時パス名は `/var/X11/renderers/DECpscf_config` ですが、このパス名は `DECPSCF_CONFIG_PATH` 環境変数を設定することにより変更できます。

#### 7.6.1.2 UDC フォント用のフォント・レンダラのセットアップ

UDC フォント・レンダラは UDC データベースに直接アクセスして、フォント・グリフを取得します。そのため、このレンダラを使用する X アプリケーションは、`cgen` ユーティリティによって作成された `.pcf` ファイルを使用する必要はありません。

UDC フォント・レンダラは、X サーバまたはフォント・サーバを介して使用するようにセットアップできます。

- X サーバの場合、フォント・レンダラはインストール時に、X サーバの構成ファイル内の `font_renderers` リストに自動的に追加される。
- フォント・サーバの場合、フォント・サーバの構成ファイル内の `renderers` リストに、次のエントリを手作業で追加しなければならない。

```
renderers = other_renderer, other_renderer,...
libfr_UDC.so;UDCRegisterFontFileFunctions
```

さらに、同じ構成ファイル内の `catalogue` リストに UDC データベースのパスを指定しなければなりません。このパスは、UDC データベースのトップ・ディレクトリに設定する必要があります。たとえば、データベースが省略時のディレクトリにセットアップされているときは、`/var/i18n/udc` がシステム全体の UDC データベースに対する適切なパスになります。

特定の言語で UDC 文字を処理するためには、次のような PostScript フォント・ディレクトリにある `fonts.dir` ファイル内に、フォント・レンダラのエントリがなければなりません。

```
/usr/i18n/lib/X11/fonts/SChinesePS
/usr/i18n/lib/X11/fonts/TChinesePS
```

`fonts.dir` ファイルを編集して、コードセットに登録されている対応する XLFD 名が後に続く、`locale_name.udc` 形式の仮想ファイル名を指定します。表 7-2 に、さまざまなアジア系コードセットに対応する XLFD エントリを示します。

表 7-2: UDC 文字用の XLFD 登録名

| コードセット                     | XLFD 登録名              |
|----------------------------|-----------------------|
| dechanyu, eucTW            | DEC.CNS11643.1986-UDC |
| big5                       | BIG5-UDC              |
| dechanzi                   | GB2312.1980-UDC       |
| deckanji, sdeckanji, eucJP | JISX.UDC-1            |

次に、`/usr/i18n/lib/X11/fonts/TChinesePS` ディレクトリにある `fonts.dir` ファイルのエントリ例を示します。

```
2
zh_TW.dechanyu.udc -system-decwin-normal-r--24-240-75-75-m-24-DEC.CNS11643.1986-UDC
zh_TW.big5.udc -system-decwin-normal-r--24-240-75-75-m-24-BIG5-UDC
```

### 7.6.1.3 TrueType フォント用のフォント・レンダラの使用

オペレーティング・システムには、TrueType フォントを使用可能にするフォント・レンダラ (`/usr/shlib/X11/libfr_TrueType.so`) が含まれています。現在、オペレーティング・システム製品に含まれている TrueType フォントは、中国語の簡体字のみです。ただし、サイトで使用するアプリケーションで別言語の他社製 TrueType フォントが必要なときには、フォント・レンダラを構成して、このようなフォントを使用することもできます。詳細は [TrueType\(5X\)](#) のリファレンス・ページを参照してください。

## 国際化インタフェースの要約表

この付録では、『X/Open CAE Specification, System Interfaces and Headers (XSH) Version 5』に規定されている、国際化インタフェース (WPI: Worldwide Portability Interfaces) の一覧と要約を示します。これらのインタフェースはすべて、ワイド文字データ型をサポートしています。またこの付録中の表には、`char` データ型を使用する古い ISO C 関数もリストされています。ただし、それらの関数では、すべての言語で文字ベースの処理を行うことはできません。各インタフェースの詳細については、リファレンス・ページ (manpages) をお読みください。XSH Version 5 に準拠する定義環境でプログラムをコンパイルする方法については、`standards(5)` のリファレンス・ページを参照してください。

### A.1 ロケール宣言

プログラムは次の関数を呼び出して、実行時に適切なロケール (言語、地域、およびコードセット) を使用します。

| WPI 関数                   | 説明                |
|--------------------------|-------------------|
| <code>setlocale()</code> | 実行時に地域化データを設定します。 |

### A.2 文字分類

以下の文字分類関数は、ロケール・カテゴリ `LC_CTYPE` で定義されているコードセットに従って値进行分类します。

| WPI 関数                  | ISO C の等価関数            | 説明                                |
|-------------------------|------------------------|-----------------------------------|
| <code>iswalnum()</code> | <code>isalnum()</code> | 文字が英数字かどうかをテストします。                |
| <code>iswalpha()</code> | <code>isalpha()</code> | 文字が英字かどうかをテストします。                 |
| <code>iswcntrl()</code> | <code>iscntrl()</code> | 文字が制御文字かどうかをテストします。               |
| <code>iswdigit()</code> | <code>isdigit()</code> | 文字が移植可能な文字セット中の 10 進数かどうかをテストします。 |

| WPI 関数                   | ISO C の等価関数             | 説明                                |
|--------------------------|-------------------------|-----------------------------------|
| <code>iswgraph()</code>  | <code>isgraph()</code>  | 文字が図形文字かどうかをテストします。               |
| <code>iswlower()</code>  | <code>islower()</code>  | 文字が英字の小文字かどうかをテストします。             |
| <code>iswprint()</code>  | <code>isprint()</code>  | 文字が印字可能かどうかをテストします。               |
| <code>iswpunct()</code>  | <code>ispunct()</code>  | 文字が区切り文字かどうかをテストします。              |
| <code>iswspace()</code>  | <code>isspace()</code>  | 文字が空白文字かどうかをテストします。               |
| <code>iswupper()</code>  | <code>isupper()</code>  | 文字が英字の大文字かどうかをテストします。             |
| <code>iswxdigit()</code> | <code>isxdigit()</code> | 文字が移植可能な文字セット中の 16 進数かどうかをテストします。 |

文字分類関数に加え、WPI にはすべての分類カテゴリに対して共通のインタフェースとなる次の関数が用意されています。

- `wctype()`  
この関数は、文字分類に対応する値を返します。
- `iswctype()`  
この関数は、ワイド文字が特定のプロパティを持つかどうかをテストします。

上記の表の WPI 関数は、次の表に示す `wctype()` と `iswctype()` 関数の呼び出しで置き換えることができます。

| 分類関数を使用する呼び出し             | <code>wctype()</code> と <code>iswctype()</code> を使用する等価な呼び出し |
|---------------------------|--------------------------------------------------------------|
| <code>iswalnum(wc)</code> | <code>iswctype(wc, wctype("alnum"))</code>                   |
| <code>iswalpha(wc)</code> | <code>iswctype(wc, wctype("alpha"))</code>                   |
| <code>iswcntrl(wc)</code> | <code>iswctype(wc, wctype("cntrl"))</code>                   |
| <code>iswdigit(wc)</code> | <code>iswctype(wc, wctype("digit"))</code>                   |
| <code>iswgraph(wc)</code> | <code>iswctype(wc, wctype("graph"))</code>                   |
| <code>iswlower(wc)</code> | <code>iswctype(wc, wctype("lower"))</code>                   |
| <code>iswprint(wc)</code> | <code>iswctype(wc, wctype("print"))</code>                   |
| <code>iswpunct(wc)</code> | <code>iswctype(wc, wctype("punct"))</code>                   |
| <code>iswspace(wc)</code> | <code>iswctype(wc, wctype("space"))</code>                   |

---

|               |                                                  |
|---------------|--------------------------------------------------|
| 分類関数を使用する呼び出し | <b>wctype()</b> と <b>iswctype()</b> を使用する等価な呼び出し |
|---------------|--------------------------------------------------|

---

|                            |                                             |
|----------------------------|---------------------------------------------|
| <code>iswupper(wc)</code>  | <code>iswctype(wc, wctype("upper"))</code>  |
| <code>iswxdigit(wc)</code> | <code>iswctype(wc, wctype("xdigit"))</code> |

---

上記の表で、`wctype()` 呼び出しに指定されている二重引用符で囲まれたリテラルは、X/Open UNIX 標準で西ヨーロッパ言語とほとんどの東ヨーロッパ言語のロケールに対して共通に定義されている文字クラスです。ただし、ロケールでは他の文字クラスを定義することもできます。Unicode 標準ではクラス固有の関数を持たない文字クラスを定義し、アジア系言語のロケールでは追加の文字クラスを定義して、表意文字と表音文字を区別できます。クラス固有の関数が存在しない場合に、ある文字があるクラスに属するかどうかをテストする場合は、`wctype` と `iswctype` 関数を使用する必要があります。XSH および Unicode 標準で定義されている文字クラスの詳細については、`locale(4)` を参照してください。

また、`isw*()` 関数の入力値は、現在のロケールで定義されているワイド文字の範囲に入っていなければなりません。入力値が範囲外の場合、その結果は不定です。詳細については、`iswctype(3)` のリファレンス・ページを参照してください。

---

#### 注意

---

上記の表の 2 番目の欄の `wctype()` 呼び出しにより、その動作が 1 番目の欄の関数と同等な `iswctype()` 呼び出しになります。ほとんどのアプリケーションでは、`wctype()` の 1 回の呼び出しに対し、`iswctype()` を複数回呼び出す必要があります。このような場合、表の 2 カラム目の関数を使用して、1 カラム目の関数と同じ処理を行うようコーディングすることもできます。

```
wctype_t    property_handle;
wint_t      wc;
int         yes_or_no;
.
.
.
    property_handle=wctype("alnum");
.
.
.
    while (...) {
        .
        .
        .
        yes_or_no=iswctype(wc, property_handle);
        .
    }
```

```
    .  
    .  
}
```

### A.3 大文字/小文字変換と汎用プロパティ変換

次の大文字/小文字変換関数を使って、ロケール・カテゴリ `LC_CTYPE` で定義されているコードセットに従い、ワイド文字の大文字/小文字変換を行います。

| WPI 関数                 | ISO C の等価関数            | 説明               |
|------------------------|------------------------|------------------|
| <code>tolower()</code> | <code>tolower()</code> | 英大文字を英小文字に変換します。 |
| <code>toupper()</code> | <code>toupper()</code> | 英小文字を英大文字に変換します。 |

また WPI には、現ロケールで定義されているプロパティに従ってワイド文字をマップしたり、変換するための関数も用意されています。

- `wctrans()`  
この関数は、ワイド文字を、現ロケールで定義されているプロパティにマップします。
- `towctrans()`  
この関数は、ワイド文字を、現ロケールで定義されているプロパティに従って変換します。

現時点では、Tru64 UNIX ロケールで定義されているプロパティは、`toupper` と `tolower` だけです。以下に `toupper()` と同等な変換処理を行う、`wctrans()` と `towctrans()` を使用したプログラム例を示します。

```
wint_t      from_wc, to_wc;  
wctrans_t   conv_handle;  
.  
.  
.  
    conv_handle=wctrans("toupper");  
.  
.  
.  
    while (...) {  
        .  
        .  
        .  
        to_wc=towctrans(from_wc,conv_handle);  
        .  
        .  
        .  
    }
```



}

## A.4 文字照合

次の WPI 関数は、LC\_COLLATE カテゴリで定義されているロケール規則に従って、ワイド文字列をソートします。

| WPI 関数   | ISO C 等価関数 | 説明         |
|----------|------------|------------|
| wscoll() | strcoll()  | 文字列を照合します。 |

A.11 節に要約されている wcsxfrm() と wcscmp() 関数を使用して、ワイド文字列を変換し比較することもできます。

## A.5 言語と文化習慣によって異なるデータへのアクセス

次の WPI 関数を使用すると、ロケール設定で指定された言語や国固有のデータをプログラムで取得できます。

| WPI 関数        | 説明                                 |
|---------------|------------------------------------|
| nl_langinfo() | ロケール設定に従って、言語や文化データを取得するための汎用関数です。 |
| strfmon()     | ロケール設定に従って、金額を書式付けます。              |
| localeconv()  | ロケール設定に従って、数値を書式付けるための情報を返します。     |

## A.6 日付/時刻の変換と書式付け

ctime() と asctime() 関数には、言語に依存しない処理を実行する柔軟性はありません。そのため、WPI にはロケール設定に従って日付と時刻の文字列を書式付ける、次の関数が用意されています。

| WPI 関数     | 説明                                                      |
|------------|---------------------------------------------------------|
| strftime() | 指定された書式文字列とロケール設定に従って、日付と時刻の文字列を書式付けます。                 |
| wcsftime() | 指定された書式文字列とロケール設定に従って日付と時刻を書式付けして、結果をワイド文字配列に返します。      |
| strptime() | 指定された書式文字列に従って、文字列を日付と時刻の値に変換します。strftime() の逆の処理を行います。 |

## A.7 テキストの書き込みと読み取り

WPI では国際化に対応するように、以下の ISO C 関数の定義を拡張しています。WPI における定義の拡張については、関数表の後で説明します。

| WPI/ISO C 関数             | 説明                                                                |
|--------------------------|-------------------------------------------------------------------|
| <code>fprintf()</code>   | <code>vararg</code> パラメータ・リストを使用して、書式付き出力をファイルに書き込みます。            |
| <code>fwprintf()</code>  | <code>vararg</code> パラメータ・リストを使用して、書式付きワイド文字を指定された出力ストリームに書き込みます。 |
| <code>printf()</code>    | <code>vararg</code> パラメータ・リストを使用して、書式付き出力を標準出力ストリームに書き込みます。       |
| <code>sprintf()</code>   | <code>vararg</code> パラメータ・リストを使用して、1 つまたは複数の値を書式付け、出力を文字列に書き込みます。 |
| <code>swprintf()</code>  | <code>vararg</code> パラメータ・リストを使用して、書式付きワイド文字を指定されたアドレスに書き込みます。    |
| <code>vfprintf()</code>  | <code>stdarg</code> パラメータ・リストを使用して、書式付き出力をファイルに書き込みます。            |
| <code>fwfprintf()</code> | <code>stdarg</code> パラメータ・リストを使用して、書式付きワイド文字を指定された出力ストリームに書き込みます。 |
| <code>vprintf()</code>   | <code>stdarg</code> パラメータ・リストを使用して、書式付き出力を標準出力ストリームに書き込みます。       |
| <code>vsprintf()</code>  | <code>stdarg</code> パラメータ・リストを書式付け、出力を文字列に書き込みます。                 |
| <code>vswprintf()</code> | <code>stdarg</code> パラメータ・リストを使用して、書式付き出力を指定されたアドレスに書き込みます。       |
| <code>vwprintf()</code>  | <code>stdarg</code> パラメータ・リストを使用して、書式付きワイド文字を標準出力に書き込みます。         |
| <code>wprintf()</code>   | <code>vararg</code> パラメータ・リストを使用して、書式付きワイド文字を標準出力に書き込みます。         |
| <code>fscanf()</code>    | ファイルからの書式付き入力を変換します。                                              |
| <code>fwscanf()</code>   | 指定された出力ストリームからの書式付きワイド文字を変換します。                                   |
| <code>scanf()</code>     | 標準入力ストリームからの書式付き入力を変換します。                                         |
| <code>sscanf()</code>    | 文字列からの書式付きデータを変換します。                                              |

| WPI/ISO C 関数           | 説明                        |
|------------------------|---------------------------|
| <code>swscanf()</code> | 指定したアドレスの書式付きワイド文字を変換します。 |
| <code>wscanf()</code>  | 標準入力からの書式付きワイド文字を変換します。   |

上記の関数群における WPI 拡張は次のとおりです。

- `%digit$` 変換指定子  
この指定子により、プリントする引数の順序を変更できます。テキストを別の言語に翻訳するときは、引数のプリント順序を変更しなければならないことが頻繁にあります。
- ロケールで定義されている小数点文字の使用  
この機能は、`e`、`E`、`f`、`g`、および `G` 変換に影響します。
- ロケールで定義されている桁区切り文字の使用
- `c` と `s` 変換文字  
これらの変換文字により、ワイド文字とワイド文字列を変換できます。

## A.8 数値変換

次の表の関数は、ワイド文字列をさまざまな数値形式に変換します。

| WPI 関数                 | ISO C の等価関数            | 説明                                                   |
|------------------------|------------------------|------------------------------------------------------|
| <code>wcstod()</code>  | <code>strtod()</code>  | ワイド文字列の先頭部分を倍精度の浮動小数点数に変換します。                        |
| <code>wcstol()</code>  | <code>strtol()</code>  | ワイド文字列の先頭部分を <code>long</code> 型の整数値に変換します。          |
| <code>wcstoul()</code> | <code>strtoul()</code> | ワイド文字列の先頭部分を <code>unsigned long</code> 型の整数値に変換します。 |

## A.9 マルチバイト文字とワイド文字の変換

アプリケーションがマルチバイト形式で外部ファイルとのデータのやり取りを行い、そのデータを内部的にワイド文字形式で処理できるようにするために、WPI には、以下の表に示すように、マルチバイト・データとワイド文字データの間の変換を行う関数が用意されています。

| WPI 関数                   | 説明                                                                                                                                                                                                                                                                                                   |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>btowc()</code>     | シングルバイトをマルチバイト文字形式からワイド文字形式に変換します。                                                                                                                                                                                                                                                                   |
| <code>mblen()</code>     | ロケール設定に従って、文字のバイト数を調べます。この関数を呼び出すには、1 バイトの文字サイズを前提としているすべての文字列処理文を変更する必要があります。以下の文は、次の文字へのポインタ <code>cp</code> を更新します。<br><br><pre>cp++;</pre> <p>使用する言語に関係なく、<code>mblen()</code> 関数が正しく動作する呼び出し例を、次に示します。MB_CUR_MAX 変数は文字を構成する最大バイト数であり、ロケールで定義されています。</p> <pre>cp += mblen(cp, MB_CUR_MAX);</pre> |
| <code>mbrlen()</code>    | <code>mblen()</code> と同じ動作をしますが、シフト状態エンコーディングを含むロケールでもリスタート可能です。 <sup>a</sup>                                                                                                                                                                                                                        |
| <code>mbrtowc()</code>   | <code>mbtowc()</code> と同じ動作をしますが、シフト状態エンコーディングを含むロケールでもリスタート可能です。 <sup>a</sup>                                                                                                                                                                                                                       |
| <code>mbstowcs()</code>  | <code>mbstowcs()</code> と同じ動作をしますが、シフト状態エンコーディングを含むロケールでもリスタート可能です。 <sup>a</sup>                                                                                                                                                                                                                     |
| <code>mbstowcs()</code>  | マルチバイト文字列をワイド文字列に変換します。                                                                                                                                                                                                                                                                              |
| <code>mbtowc()</code>    | マルチバイト文字をワイド文字に変換します。                                                                                                                                                                                                                                                                                |
| <code>wctombs()</code>   | ワイド文字列をマルチバイト文字列に変換します。                                                                                                                                                                                                                                                                              |
| <code>wcrtomb()</code>   | <code>wctomb()</code> と同じ動作をしますが、シフト状態エンコーディングを含むロケールでもリスタート可能です。 <sup>a</sup>                                                                                                                                                                                                                       |
| <code>wcsrtombs()</code> | <code>wctombs()</code> と同じ動作をしますが、シフト状態エンコーディングを含むロケールでもリスタート可能です。 <sup>a</sup>                                                                                                                                                                                                                      |
| <code>wctob()</code>     | 可能であれば、ワイド文字をマルチバイト形式のシングルバイト文字に変換します。                                                                                                                                                                                                                                                               |
| <code>wctomb()</code>    | ワイド文字をマルチバイト文字に変換します。                                                                                                                                                                                                                                                                                |

<sup>a</sup>本オペレーティング・システムは現在、シフト状態エンコーディングを使用するロケールをサポートしていません。

## 注意

ファイル・コード (マルチバイト・データ) の明示的な変換処理が常に必要なわけではありません。テキストの書き込みと読み

込みを行う関数群 (A.7 節で説明) には書式指定 %s と %C が含まれるため、マルチバイト文字からワイド文字への変換は自動的に処理されます。古い ISO C の入出力関数 (A.10 節で説明) に対応する WPI 関数も、マルチバイト文字からワイド文字への変換を自動的に処理します。

## A.10 入出力

次の表にリストされている WPI 関数は、テキストの入出力操作のために、ファイル・コード (一般にマルチバイト・エンコーディング) とプロセス・コード (ワイド文字エンコーディング) 間の変換を自動的に行います。

| WPI 関数     | ISO C の等価関数 | 説明                                                                        |
|------------|-------------|---------------------------------------------------------------------------|
| fgetwc()   | fgetc()     | 入力ストリームから文字を取得して、ワイド文字に変換します。                                             |
| fgetws()   | fgets()     | 入力ストリームから文字列を取得して、ワイド文字列に変換します。                                           |
| fputwc()   | fputc()     | ワイド文字をマルチバイト文字に変換し、結果を出力ストリームに書き込みます。                                     |
| fputws()   | fputs()     | ワイド文字列をマルチバイト文字列に変換し、結果を出力ストリームに書き込みます。                                   |
| fwide()    | なし          | ストリーム・オリエンテーションをバイトまたはワイド文字に設定します。この関数を現在のロケール環境で使用しても無意味です。 <sup>a</sup> |
| getwc()    | getc()      | 引数として関数に渡される文字を入力ストリームから取得して、ワイド文字に変換します。                                 |
| getwchar() | getchar()   | 標準入力ストリームから文字を取得して、ワイド文字に変換します。                                           |
| なし         | gets()      | fgetws() を参照してください。                                                       |
| mbsinit()  | なし          | シフト状態エンコーディングを使用するロケールで、マルチバイト文字列が初期変換状態であるかどうかを判別します。 <sup>a</sup>       |
| putwc()    | putc()      | ワイド文字をマルチバイト文字に変換し、結果を出力ストリームに書き込みます。変換された文字は、引数として関数に渡されます。              |

| WPI 関数       | ISO C の等価関数 | 説明                                      |
|--------------|-------------|-----------------------------------------|
| putwchar ( ) | getchar ( ) | ワイド文字をマルチバイト文字に変換し、結果を標準出力ストリームに書き込みます。 |
| なし           | puts ( )    | fputws ( ) を参照してください。                   |
| ungetwc ( )  | ungetc ( )  | ワイド文字を入力ストリーム上にプッシュバックします。              |

<sup>a</sup>本オペレーティング・システムは現在、シフト状態エンコーディングを使用するロケールをサポートしていません。

## A.11 文字列処理

WPI ではワイド文字列の操作をサポートするために、ISO C 文字列処理関数の代替関数と新規の関数を規定しています。WPI 関数は、シングルバイト文字とマルチバイト文字の両方をサポートします。

### 文字列の連結

| WPI 関数      | ISO C の等価関数 | 説明                                                     |
|-------------|-------------|--------------------------------------------------------|
| wcscat ( )  | strcat ( )  | 文字列を他の文字列の終わりにコピーします。                                  |
| wcsncat ( ) | strncat ( ) | コピーする文字数がパラメータ <i>n</i> によって制限される以外は、wcscat ( ) と同じです。 |

### 文字列の検索

| WPI 関数      | ISO C の等価関数 | 説明                                                                              |
|-------------|-------------|---------------------------------------------------------------------------------|
| wcschr ( )  | strchr ( )  | ワイド文字列中で最初に現れるワイド文字を探します。                                                       |
| wcsrchr ( ) | strrchr ( ) | ワイド文字列中で最後に現れるワイド文字を探します。                                                       |
| wcspbrk ( ) | strpbrk ( ) | ある文字列に含まれる文字が、他方の文字列で最初に現れる位置を探します。                                             |
| wcsstr ( )  | strstr ( )  | ワイド文字列中の部分文字列を探します。wcsstr ( ) は、Issue 5 以前の XSH 仕様で規定されている wcswcs ( ) に置き換わります。 |

| WPI 関数   | ISO C の等価関数 | 説明                                                     |
|----------|-------------|--------------------------------------------------------|
| wscspn() | strcspn()   | 1 番目のワイド文字列の中で、2 番目のワイド文字列に含まれていない文字で構成する先頭部分の長さを返します。 |
| wcsspn() | strspn()    | 1 番目のワイド文字列の中で、2 番目のワイド文字列に含まれる文字で構成する先頭部分の長さを返します。    |

## 文字列のコピー

| WPI 関数    | ISO C の等価関数 | 説明                                                           |
|-----------|-------------|--------------------------------------------------------------|
| wscpy()   | strcpy()    | ワイド文字列をコピーします。                                               |
| wcsncpy() | strncpy()   | コピーするワイド文字の数がパラメータ <i>n</i> によって制限されること以外は、strcpy() 関数と同じです。 |

## 文字列の比較

| WPI 関数    | ISO C の等価関数 | 説明                                                          |
|-----------|-------------|-------------------------------------------------------------|
| wscmp()   | strcmp()    | 2 つのワイド文字列を比較します。                                           |
| wcsncmp() | strncmp()   | 比較対象のワイド文字数がパラメータ <i>n</i> によって制限されること以外は、strcmp() 関数と同じです。 |

## 文字列長の判定

| WPI 関数   | ISO C の等価関数 | 説明                  |
|----------|-------------|---------------------|
| wcslen() | strlen()    | ワイド文字列中の文字の数を判定します。 |

## 文字列の分割

| WPI 関数   | ISO C の等価関数 | 説明                                             |
|----------|-------------|------------------------------------------------|
| wcstok() | strtok()    | ワイド文字列をトークンの集合に分割します。各トークンは、指定されたワイド文字で区切られます。 |

## 出力位置の判定

| WPI 関数     | ISO C の等価関数 | 説明                                |
|------------|-------------|-----------------------------------|
| wcswidth() | なし          | ワイド文字列中の文字を出力するのに必要なポジション数を判定します。 |
| wcwidth()  | なし          | ワイド文字を出力するのに必要なポジション数を判定します。      |

## ワイド文字列上でのメモリ操作

| WPI 関数     | ISO C の等価関数 | 説明                                                |
|------------|-------------|---------------------------------------------------|
| wmemcpy()  | memcpy()    | あるバッファから別のバッファにワイド文字をコピーします。                      |
| wmemchr()  | memchr()    | バッファ内で、指定されたワイド文字を探します。                           |
| wmemcmp()  | memcmp()    | 2つのバッファ内で、指定された数のワイド文字を比較します。                     |
| wmemmove() | memmove()   | 重なっている部分に悪影響を及ぼすことなく、あるバッファから他のバッファにワイド文字をコピーします。 |
| wmemset()  | memset()    | 指定された数のワイド文字でバッファを埋めます。                           |

## A.12 コードセット変換

WPI にはコードセット変換機能が用意されています。プログラム中で変換機能を利用するには関数を使用し、対話形式で利用するには `iconv` コマンドを使います。プログラムやコマンド・レベルで、ソース・コードセットおよびターゲット・コードセットと、変換対象の言語テキスト・ファイルの名前を指定します。コードセットは、言語テキストが渡される変換ストリームを定義します。

次の表に、コードセット変換に使用する関数の要約を示します。これらの関数は、ライブラリ `libiconv.a` に含まれます。



| WPI 関数                     | ISO C の等価関数 | 説明                                                          |
|----------------------------|-------------|-------------------------------------------------------------|
| <code>iconv_open()</code>  | なし          | ソース・コードセットとターゲット・コードセットを判定し、変換ストリームを初期化します。                 |
| <code>iconv_close()</code> | なし          | 変換ストリームをクローズします。                                            |
| <code>iconv()</code>       | なし          | ソース・コードセットでエンコードされた入力文字列を、ターゲット・コードセットでエンコードされた出力文字列に変換します。 |

`iconv` コマンドの説明と、サポートされている変換タイプについては、7.5 節を参照してください。`iconv` ライブラリとプログラムの使用については、`iconv(3)` のリファレンス・ページを参照してください。



# B

## ユーザ定義文字データベースのセットアップと使用法

日本語，中国語，韓国語では，アジア系言語の標準の文字セットで定義されていない文字を補うために，ユーザ定義文字 (UDC) を使用することができます。この付録では，UDC と，UDC の入力および表示をサポートするファイルの作成方法について説明します。

UDC の作成には，B.1 節で説明する `cedit` エディタを使用します。UDC のフォント・ファイル，照合ファイル，およびその他のサポート・ファイルを作成するには，B.2 節で説明する `cgen` コマンドを使用します。X アプリケーションはフォント・レンダラを使用して，UDC データベースから直接 UDC 用のフォントを取得できます。フォント・レンダラについては，7.6 節を参照してください。

### 注意

システム標準の `sort` コマンドは，UDC 用に作成された照合ファイルにはアクセスしません。これらのファイルにアクセスするには，`asort` コマンドを使用します。これらの文字を含む文字列のソートについては，`asort(1)` のリファレンス・ページと，『国際化機能ユーザズ・ガイド』を参照してください。

端末あるいはワークステーションのモニタで UDC を表示させるためには，そのための設定操作が必要です。

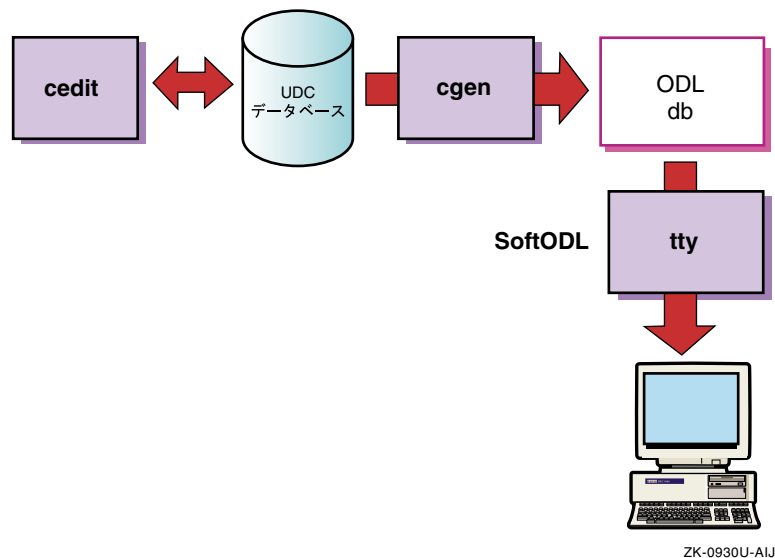
`atty` ドライバには，UDC に関連するファイルのオンデマンド・ローディングを可能にするメカニズムが備わっています。このメカニズムを有効にすると，`stty` コマンドで省略時のパラメータ値を変更できます。表 B-1 に，オンデマンド・ローディングで使用する `stty` コマンド・オプションを示します。

表 B-1: UDC サポート・ファイルのオンデマンド・ローディングのための **stty** オプション

| stty オプション   | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| odl          | ソフトウェア・オンデマンド・ローディング (SoftODL) サービスを有効にします。                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| -odl         | ソフトウェア・オンデマンド・ローディング (SoftODL) サービスを無効にします。                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| odlsize size | ODL バッファの最大サイズを設定します。このサイズは、端末のフォント・キャッシュ・サイズと同じでなければなりません。省略時の <i>size</i> は 256 文字です。                                                                                                                                                                                                                                                                                                                                                                                               |
| odltype type | ODL バッファの置換方式を設定します。 <i>type</i> に指定できる値は、 <i>fifo</i> (先入れ先出し) と <i>lru</i> (最低使用頻度) です。                                                                                                                                                                                                                                                                                                                                                                                             |
| odldb path   | UDC をサポートするデータベースとその他のファイルのパスを設定します。<br><br>このパスが指定されていない場合、システムの省略時ファイルが使用されるか、あるいは、ユーザが個人用の UDC データベースを作成できる場合は、プロセスの省略時ファイルが使用されます。<br><br>各種データベースの省略時パス名は、この章の後半で説明するファイル <code>/var/i18n/conf/cp_dirs</code> で指定されます。たとえば、 <code>cp_dirs</code> ファイルにはシステム全体の省略時パス名が <code>/var/i18n/udc</code> と <code>/var/i18n/odl</code> であり、プロセスの省略時パス名が <code>\$HOME/.udc</code> と <code>\$HOME/.odl</code> であることを指定できます。 <code>odl</code> ファイルを変更したいときは、 <code>odldb</code> オプションを使用します。 |
| odlreset     | ODL サービスをリセットし、内部 ODL バッファをクリアします。                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| odlall       | ODL サービスの現在の設定を表示します。                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

図 B-1 に、表 B-1 で説明したコンポーネントと SoftODL サービスの関係を示します。

図 B-1: ユーザ定義文字をサポートするコンポーネント



## B.1 ユーザ定義文字の作成

UDC エディタ (cedit コマンドで起動されます) は、ユーザ定義文字の属性を管理するための curses アプリケーションです。cedit で操作できる文字属性は次のとおりです。

- シンボリック名
- ビットマップ・フォント用のスタイルとサイズ (16x18 , 24x24 , 32x32 , および 40x40)
- コードセット値
- 照合値
- サポートされている入力システムの入力キー・シーケンス
- 文字クラス

ユーザ定義文字には、文字属性 UDC データベースに格納される、それぞれの文字属性レコードがあります。UDC データベースには、システム全体で使用されるものと、個人用のものがあります。全ユーザが共有するシステム全体のデータベースは1つしか設定できませんが、すべてのユーザがシステム全体のデータベースの他に個人用のデータベースを持つことができます。

UDC エディタを起動するには、次のように入力します。

% cedit

オプションを指定しないで cedit コマンドを実行すると、省略時のデータベースが使用されます。スーパーユーザの場合、省略時のデータベースは /var/i18n/udc になります。一般ユーザの場合には、省略時のデータベースは \$HOME/.udc になります。個人用データベース内の UDC を使用した場合、さまざまな問題が発生する可能性があります。たとえば、個人用データベースに定義されている属性に依存している文字を使用したデータを交換するユーザは、個人用のデータベースを共同で管理しなければなりません。このような問題を回避するには、特権ユーザが、システム全体のデータベースですべての UDC を管理するようにしてください。表 B-2 に示すように、credit コマンドには多数のオプションと引数があります。

表 B-2: credit コマンドのオプション

| credit のオプションと引数 | 説明                                                                                                                                                                                   |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -c <i>old_db</i> | 日本語 ULTRIX の fedit フォント・ファイルやアジア版 ULTRIX の文字属性データベース・ファイルを、credit で使用されるフォーマットに変換します。                                                                                                |
| <i>cur_db</i>    | 文字属性データベースのパスを指定します (省略時のパスを無効にします)。                                                                                                                                                 |
| -h               | credit の構文を表示します。                                                                                                                                                                    |
| -r <i>ref_db</i> | 基準文字属性データベースのパスを指定します (省略時のパスを無効にします)。<br>このデータベースは、credit で操作する UDC データベースのモデルとして使用します。<br>credit の [File] メニューの Reference Database 項目は、credit のコマンド行で -r オプションを指定した場合に得られる結果と同じです。 |

credit コマンドは、UDC の作成をサポートしていないロケール設定の場合、エラー・メッセージを返します。UDC がサポートされているロケールには、中国語や日本語のロケールがあります。credit を起動した後に、credit ユーザ・インタフェース画面の [Options] メニューを使用して、ユーザ・インタフェース・メッセージやヘルプ・テキストを英語に戻すことができます。

## 注意

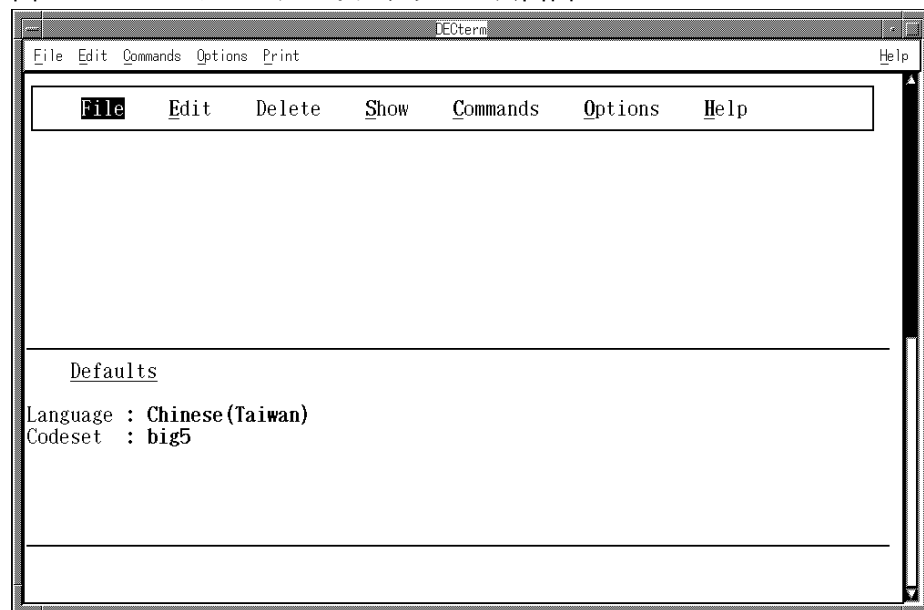
dtterm 端末エミュレータは、`cedit` の機能をサポートしていません。このため、`dtterm` 下で `cedit` を使用しようとする、UDC マネージャ・ユーティリティがハング・アップすることがあります。`cedit` の機能をサポートしている、`dxterm` 端末エミュレータを使用してください。

次の項では、`cedit` の画面、メニュー項目、編集モード、およびファンクション・キーについて説明します。

### B.1.1 `cedit` のユーザ・インタフェース画面上での操作

`LANG` 変数が `zh_TW.big5` などのサポートされているロケールに設定されている場合、`cedit` コマンドは、ユーザ・インタフェース画面を表示します (図 B-2)。

図 B-2: `cedit` のユーザ・インタフェース画面



ZK-0924U-R

ユーザ・インタフェース画面は、次の領域に分割されています。

- メニュー領域

この領域には、メニュー・バーが含まれます。特定のメニューを選択して起動すると、その項目がメニュー・バーの下メニュー領域に表示されます。

- ステータス領域

メニュー領域の下にはステータス領域があり、現在の言語とコードセットが表示されます。

- 入力およびメッセージ領域

画面下部の 2 行は、ユーザが入力を行ったり、警告メッセージや情報メッセージが表示される領域です。

メニュー上の項目を参照するには、メニュー・タイトルの下線が付いている文字のキーを押します。または、キーボードの 4 つの矢印キーを使用してメニューを選択し、Return キーまたはスペース・バーを押します。

メニュー項目は、次のいずれかの状態で表示されます。

- アクティブ

アクティブな項目とは、ユーザが選択できる項目のことです。アクティブな項目では、1 つの文字が強調表示され、下線が付けられています。その文字のキーを押すと、その項目の機能を起動できます。

- 非アクティブ

非アクティブな項目は選択することができません。非アクティブな項目には、下線付きの強調表示された文字は含まれません。

- 選択

強調表示されたキーではなく、下矢印キーを押すと、項目の表す機能を起動せずに、その項目を選択できます。現在選択されている項目は、反転表示されます。

- 起動

強調表示された文字のキーを押すか、あるいは下矢印キーにより項目を選択した後で Return キーかスペース・バーを押すと、項目を起動できます。一般に、項目を起動したときは、ポップアップ・メニューが表示されるか、特定の機能が実行されるか、あるいは、その両方が実行されます。 >> が後に付いている項目を起動すると、カスケード・メニューが表示されます。



項目を起動せずに、1 つ上のレベルのメニューに戻るには、Ctrl/X を押します。

ユーザ・インタフェース画面のメニューには、ユーザ定義文字とその属性を管理するための、次のようなオプションがあります。

- File

[File] メニューを使用すると、次の操作が行えます。

- 現在の文字に加えた変更を保存する。
- 現在処理している文字に加えた変更を取り消す。
- 基準文字属性データベースを変更する。
- cedit プログラムを終了する。

- Edit

[Edit] メニューを使用すると、文字を選択し、その文字のフォント・グリフ、コードセット値、照合値、入力キー・シーケンス、クラス、名前などを作成し、変更できます。

B.1.2 項に、文字のフォント・グリフの編集方法が解説されています。コードセットや照合値の変更、入力キー・シーケンスについては、この項の [Show]、[Commands]、[Options] のメニューの説明を参照するか、cedit(1) のリファレンス・ページを参照してください。

- Delete

[Delete] メニューを使用すると、文字やその属性を削除できます。

- Show

[Show] メニューを使用すると、現在処理している文字の属性や、データベース (現在の文字属性データベースまたは基準文字属性データベース) のステータスを表示できます。

cedit ユーティリティは、属性レコードを使用して文字をトラックします。このレコードには、次の属性を識別するためのフィールドが含まれています。

- 文字番号 (UDC データベース内の文字ごとに一意の番号)
- コードセット値 (特定の言語と地域の組み合わせでサポートされているコードセットごとに 1 つの値)
- フォント・スタイルとサイズ

- 照合値 (言語でサポートされる照合シーケンスごとに 1 つの値)
- 入力キー・シーケンス (言語でサポートされる語句入力方式ごとに 1 つのシーケンス)
- クラス識別子 (将来の使用のために予約されている)
- 文字ニーモニック (将来の使用のために予約されている)

アジア系言語のコードセット間には、UDC 属性のサポートに関していくつかの違いがあります。たとえば `cedit` では、日本語のユーザ定義文字の入力キー・シーケンスを定義することはできません。中国語では、DEC Hanyu コードセットと、TsangChi および QuickTsangChi 入力モードを組み合わせて使用する場合にのみ、入力キー・シーケンスを定義できます。

- **Commands**

[Commands] メニューを使用すると、次の操作が行えます。

- 文字レコードを、基準文字属性データベースから現在の文字属性データベースにコピーする。あるいは、現在の文字属性データベース内で、ある範囲の文字レコードを別の範囲にコピーする。

コピー操作には、確認しない (No Confirm)、範囲内のすべての文字のコピー操作を確認する (ConfirmAll)、あるいは他の文字を上書きするコピー操作の場合にのみ確認する (Confirm Conflict) の 3 種類のコピー方法があります。

- 現在の言語およびコードセット設定において、現在の文字属性データベース内で定義されているすべての文字を表示する。
- 文字のフォント・サイズを変更する。

文字をあるフォント・サイズに設定した後でこのオプションを使用すると、他のサイズのフォントを作成できます。スケーリング・アルゴリズムが単純なものであるため、スケーリングを行った後で手作業によりフォント・グリフを修正しなければならないことがあります。

- **Options**

[Options] メニューを使用すると、UDC の処理作業に使用する言語とコードセットの現在の設定を変更できます。また、`cedit` のユーザ・インタフェースで表示されるメッセージとヘルプ・テキストの言語を別々

に設定することもできます。省略時の `cedit` ユーザ・インタフェースの言語は、`cedit` を起動したときのロケール設定と同じです。

- Help

[Help] メニューを使用すると、`cedit` の機能についての説明が表示されます。また、キーボードに Help キーがあれば、あるいは、ワークステーション・ユーザの場合には端末設定で Help キーが有効になっていれば、Help キーを使用してメニュー項目についてのヘルプも表示できます。矢印キーでメニュー項目を選択し、次に Help キーを押すと、選択された項目の簡単な説明が表示されます。

## B.1.2 フォント・グリフの編集

ユーザ定義文字のフォント・グリフを作成または変更するには、以下の手順に従って、`cedit` のフォント編集画面を呼び出さなければなりません。

1. [Edit] メニューから [Character] 項目を選択し、文字を選択します。

`cedit` は、編集する文字の 16 進コード値 (`\x` プレフィックスは付けません) を入力するように求めます。UDC 文字の有効なコード範囲は、一連の構成ファイルで定義されています。現在のロケールの言語と地域に対して、複数のコードセットがサポートされている場合、`cedit` は、関連するすべてのロケールでその文字が使用できるように、その他のコードセットの値も判定しようとします。

`cedit` が、他のコードセットにおける文字の値を判定できない場合には、[Options] メニューでコードセット設定を変更し、そのコードセットにおける文字のコードを明示的に指定することもできます。

一般に、その言語でサポートされている他のコードセットに対してマップ可能な値を UDC が持つように定義します。特定のアジア系言語における UDC のコードについては、次の言語固有のマニュアルを参照してください。

- 『*Technical Reference for Using Chinese Features*』
- 『日本語機能ガイドブック』
- 『*Technical Reference for Using Korean Features*』
- 『*Technical Reference for Using Thai Features*』

これらのマニュアルは、本オペレーティング・システムのドキュメント Web サイト ([http://www.tru64unix.com-paq.com/docs/pub\\_page/V51A\\_DOCS/PRG\\_DOCS.HTM](http://www.tru64unix.com-paq.com/docs/pub_page/V51A_DOCS/PRG_DOCS.HTM)) で、プログラミング関連ドキュメントとして掲載されています。

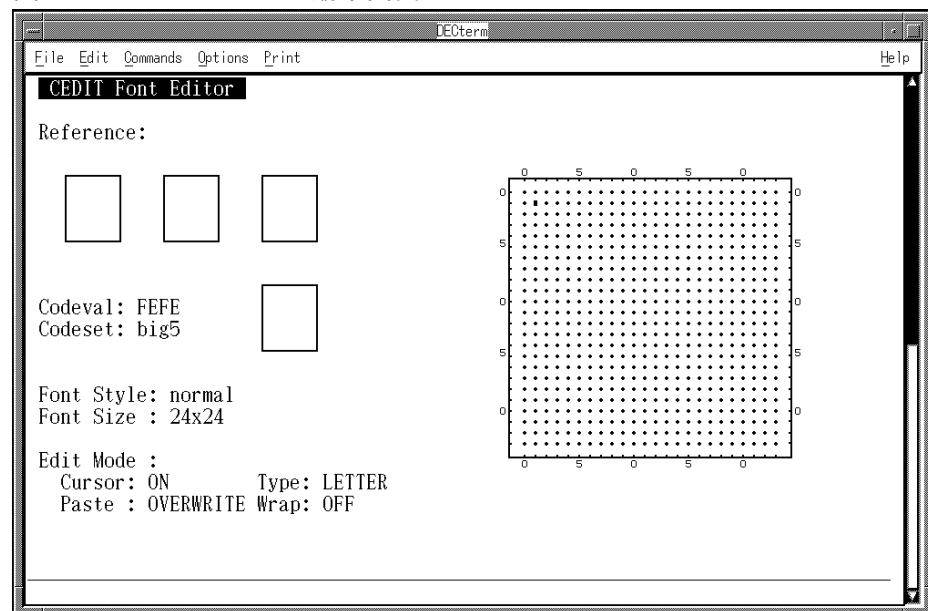
cedit エディタはまず、ユーザが入力したコードを現在の UDC データベースで検索します。そのコードの文字が UDC データベースになければ、現在の基準文字データベースで検索します。

2. [Edit] メニューの Font 項目を選択し、フォント・スタイル/サイズのオプションを表示します。
3. フォント・スタイル/サイズ・オプションのいずれかを選択します。

Motif アプリケーションで使用するフォント・グリフを作成する場合、使用可能なサイズ・オプションが、そのフォントを使用するウィンドウ領域に適していないことがあります。その場合には、フォントの縦横両方の大きさに対応できる一番小さなサイズ・オプションを選択します。

cedit エディタは、フォントの全画面エディタ・インタフェースを表示します (図 B-3)。

図 B-3: cedit フォント編集画面



ZK-0925U-R

cedit のフォント編集画面には、次のウィンドウがあります。

- 画面右側の大きなウィンドウは、UDC フォント・グリフを編集するためのウィンドウです。編集には、cedit のカーソル移動機能と編集機能を使用します。  
編集ウィンドウ上の個々のドットは 1 ピクセルを表します。
- Reference タイトルのすぐ下の 3 つの小さなウィンドウには、現在のフォント・グリフを編集するときに参照可能な他のフォント・グリフが表示されます。これらのウィンドウに表示されるフォント・グリフは、cedit の Refer 機能を使用して制御できます。
- 3 つの参照ウィンドウの下にある小さなウィンドウは、表示ウィンドウと呼ばれます。表示ウィンドウは、編集中のフォント・グリフを実寸大で表示します。表示ウィンドウは、編集ウィンドウで行った変更を自動的に反映しません。表示ウィンドウのフォント・グリフを更新するには、KP キーを押さなければなりません。

---

#### 注意

---

小さなウィンドウでのフォント・グリフ表示には、ハードウェア上の制約がいくつかあります。

参照および表示ウィンドウでのフォント・グリフの表示は、一部の端末、特に動的置換文字セット (DRCS) 機能をサポートしているローカル言語端末でのみ可能です。

端末エミュレーション・ウィンドウでは、表示ウィンドウのフォント・グリフは実寸大では表示されません。

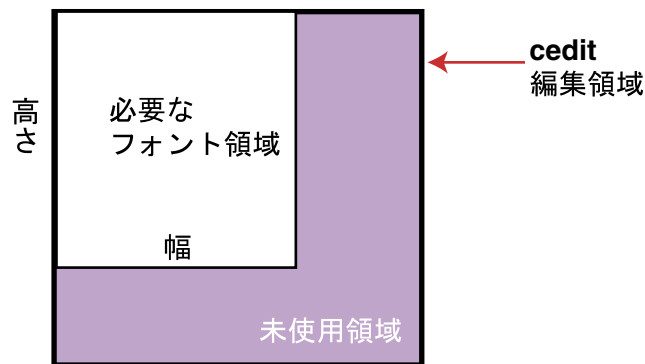
---

編集ウィンドウで作成した、システム・ソフトウェア用のフォントは、エディタ・インタフェース画面が表示される前に、ユーザが選択したサイズになるように処理されます。

また、Motif アプリケーション用のフォントを作成することもできますが、Motif アプリケーション用のフォントは選択したサイズよりも小さくなります。この場合、編集操作は、編集ウィンドウの左上隅から始まる長方形内で行い、利用可能な編集領域よりも小さいサイズのフォントに制限されます (図 B-4 を参照)。

Motif アプリケーションをサポートする UDC フォント・コンバータは、編集ウィンドウの左上隅をフォントの原点と見なし、この原点に基づいてグリフを囲むのに必要な大きさを決定し、その外側にある未使用の領域を破棄します。また、このユーティリティを使用すると、コンパイル済みのフォント・グリフのサイズを明示的に指定することもできます。

図 B-4: フォント・サイズを変更するフォント編集画面の解釈  
原点



ZK-0932U-AI J

cedit のすべての機能はキーにバインドされています。つまり、キーを押すことにより、機能を実行できます。PF2 キーまたは Help キーを押すと、各キーがバインドされている編集機能のダイアグラムが表示されます。システムによってキーパッドのデザインが異なるため、オンライン・ダイアグラムがこの項の説明と異なることもあります。cedit 編集画面には、次の編集モードがあります。

- カーソル・モード

矢印キーを使用してカーソルを移動しても、ピクセル状態に影響することはありません。ただし、キーパッド・キーを使用してカーソルを移動したときは、カーソル・モードはピクセル状態に次のような影響を与えます。

- On: カーソルの下のピクセルをオンにする。
- Off: カーソルの下のピクセルをオフにする。
- On/Off: カーソルの下のピクセルを反転する。

また、KP5 キーを押すことにより、カーソルの下のピクセルを反転することもできます。

- Move: ピクセルの状態を変更せずにカーソルを移動する。

- ペースト・モード

ペースト・モードは、ペースト機能を実行したときのピクセルの動作を制御します。

- Overlay: ペースト・バッファ内の対応するピクセルがオンになっている場合に、ピクセルをオンにする。
- Overwrite: ピクセルを、ペースト・バッファの対応するピクセルの状態に設定する。

- タイプ・モード

タイプ・モードは、文字の周囲に 1 ピクセル幅のマージンを置くかどうかを判定します。

- Body: フォント・グリフ領域全体の編集を可能にする。
- Letter: 境界領域のピクセル値の編集を不可能にする。このモードでは、編集ウィンドウの境界にあるピクセルの状態をオンにすることはできません。

- ラップ・モード

ラップ・モードは、カーソル・ラッピングをオンまたはオフにします。

- On: 編集領域の右端のピクセルを越えてカーソルを移動すると、カーソルは左端のピクセルに折り返す。

編集領域の左端、上端、あるいは下端のピクセルを越えてカーソルを移動したときも、同じような折り返しが起こります。

- Off: 編集領域の左端、右端、上端、あるいは下端のピクセルを越えてカーソルを移動しようとする、警告音が鳴って、カーソル移動が停止する。

`cedit` は、4 つのバッファを使用してビットマップ・データを保持します。これらのバッファのいくつかは、バッファの説明の次に解説する編集機能で使用されます。

- 編集バッファ

通常このバッファの内容は、編集ウィンドウに表示されます。

- 使用バッファ

このバッファは Use 機能に関連付けられており、UDC データベースや、参照ウィンドウの 1 つから取り出したフォント・グリフを含みます。

- カット-アンド-ペースト・バッファ

このバッファは、ビットマップ・データを編集ウィンドウにペーストするときに使用します。ペーストされるビットマップ・データは、使用バッファまたは編集バッファからコピーされます (編集ウィンドウの一部を別の部分にコピーする場合)。

- 取り消しバッファ

このバッファには、最後の編集操作で加えられた変更が含まれており、cedit の Undo 機能がそれらの変更を取り消すのに使用します。

フォント編集画面上でウィンドウ操作を行っているときは、キーストロークを使用するか、場合によっては Do キーを押したときに表示されるポップアップ・メニューを使用して、編集機能呼び出します。ポップアップ・メニューでは、次の機能を使用できます。

- Scale

この機能を使用すると、現在のフォント・グリフを、システムでサポートされている他のサイズに変更できます。SCALE 機能には対応するキーストロークがないため、ポップアップ・メニュー以外では使用できません。

- Use

この機能は、フォント・グリフを UDC データベースから、あるいは参照ウィンドウの 1 つから取り出します。

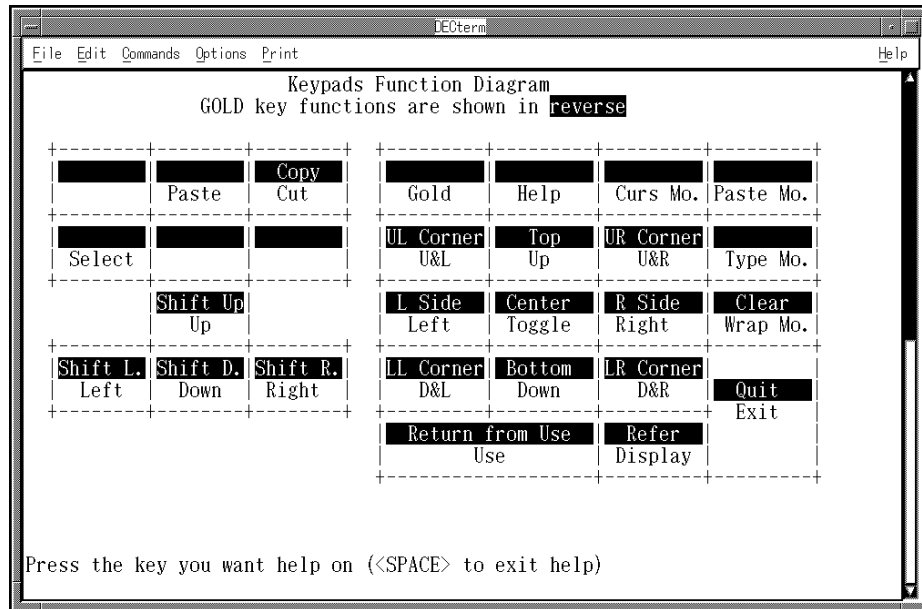
- Refer

この機能は、UDC データベースからコピーしたフォント・グリフを、参照ウィンドウの 1 つにコピーします。

図 B-5 に、さまざまな編集機能呼び出すためのキーパッド・キーマップを示します。その次の表には、キーパッド機能と、描画に使用される英字キーの説明を示します。



図 B-5: cedit の編集機能呼び出すためのキーマップ



ZK-0926U-R

表 B-3: さまざまなフォント編集機能のためのキー

| キー           | 説明                                                                                                                            |
|--------------|-------------------------------------------------------------------------------------------------------------------------------|
| Help または PF2 | キーと編集機能の対応付けを表示します。特定のキーの編集機能については、図の中のキーと Help キーを同時に押してください。                                                                |
| PF1          | GOLD 状態 (代替機能を持つキーに関するワード・プロセッサ用語) を切り替えます。一部のキーボード・キーには、複数の機能が対応付けられています。そのような場合は、PF1 キーを押してから他のキーボード・キーを押すと、機能の 1 つが実行されます。 |
| KP.          | 表示ウィンドウ上に実寸大のフォント・グリフを表示します。                                                                                                  |
| GOLD KP.     | 編集ウィンドウ上に表示されているフォント・グリフをクリアします。                                                                                              |
| U または u      | 直前の操作を取り消します。                                                                                                                 |
| Ctrl/L       | 画面を再描画します。                                                                                                                    |
| Ctrl/Z       | cedit を停止します。                                                                                                                 |
| Do           | SCALE, USE, および REFER 機能呼び出すためのポップアップ・メニューを表示します。                                                                             |

表 B-3: さまざまなフォント編集機能のためのキー (続き)

| キー         | 説明                        |
|------------|---------------------------|
| Enter      | 変更を保存し、フォント・エディタを終了します。   |
| GOLD Enter | 変更を保存せずに、フォント・エディタを終了します。 |

表 B-4: cedit モードを切り替えるためのキー

| キー  | 説明               |
|-----|------------------|
| PF3 | カーソル・モードを切り替えます。 |
| PF4 | ペースト・モードを切り替えます。 |
| KP- | タイプ・モードを切り替えます。  |
| KP. | ラップ・モードを切り替えます。  |

表 B-5: カーソル移動を制御するためのキー

| キー  | 説明                            |
|-----|-------------------------------|
| 上矢印 | カーソルを上に移動します。                 |
| 下矢印 | カーソルを下に移動します。                 |
| 左矢印 | カーソルを左に移動します。                 |
| 右矢印 | カーソルを右に移動します。                 |
| KP7 | カーソル・モードに応じて、カーソルを左上に移動します。   |
| KP8 | カーソル・モードに応じて、カーソルを上に移動します。    |
| KP9 | カーソル・モードに応じて、カーソルを右上に移動します。   |
| KP4 | カーソル・モードに応じて、カーソルを左に移動します。    |
| KP6 | カーソル・モードに応じて、カーソルを右に移動します。    |
| KP1 | カーソル・モードに応じて、カーソルを左下に移動します。   |
| KP2 | カーソル・モードに応じて、カーソルを下に移動します。    |
| KP3 | カーソル・モードに応じて、カーソルを右下に移動します。   |
| KP5 | カーソルを移動せずに、カーソルの下のピクセルを反転します。 |

表 B-6: ウィンドウ領域にカーソルを移動するためのキー

| キー <sup>a</sup> | 説明                   |
|-----------------|----------------------|
| GOLD KP7        | カーソルを左上隅に移動します。      |
| GOLD KP8        | カーソルを一番上の行に移動します。    |
| GOLD KP9        | カーソルを右上隅に移動します。      |
| GOLD KP4        | カーソルを一番左のカラムに移動します。  |
| GOLD KP5        | カーソルをウィンドウの中央に移動します。 |
| GOLD KP6        | カーソルを一番右のカラムに移動します。  |
| GOLD KP1        | カーソルを左下隅に移動します。      |
| GOLD KP2        | カーソルを一番下の行に移動します。    |
| GOLD KP3        | カーソルを右下隅に移動します。      |

<sup>a</sup>PF1 キーは、GOLD 状態を切り替えます。

表 B-7: フォント・グリフを描画するためのキー

| キー      | 説明                           |
|---------|------------------------------|
| L または l | 選択した 2 つの点を結ぶ線を描きます。         |
| C または c | 選択した点を中心とする円を描きます。           |
| r       | 選択した領域に、点線の長方形を描きます。         |
| R       | 選択した領域に、実線の長方形を描きます。         |
| e       | 選択した領域に、点線の楕円を描きます。          |
| E       | 選択した領域に、実線の楕円を描きます。          |
| X または x | 横軸 (X 軸) に沿ってフォント・グリフを反転します。 |
| Y または y | 縦軸 (Y 軸) に沿ってフォント・グリフを反転します。 |
| /       | 45 度の対角線に沿ってフォント・グリフを反転します。  |
| \       | 135 度の対角線に沿ってフォント・グリフを反転します。 |
| F または f | カーソル・モードに応じて、領域を塗り潰します。      |
| T または t | すべてのピクセル状態を反転させます。           |

表 B-8: フォント・グリフを編集するためのキー

| キー <sup>a</sup> | 説明                                                     |
|-----------------|--------------------------------------------------------|
| KP0             | 編集ウィンドウの表示を、編集バッファ内のフォント・グリフから、使用バッファ内のフォント・グリフに変更します。 |
| GOLD KP.        | 参照ウィンドウにフォント・グリフを表示します。                                |
| GOLD KP0        | 編集ウィンドウの表示を、使用バッファ内のフォント・グリフから、編集バッファ内のフォント・グリフに変更します。 |
| Select          | 選択した領域で操作を開始するか、取り消します。                                |
| Insert          | カット-アンド-ペースト・バッファの内容を挿入します。                            |
| Remove          | 選択した領域をカット-アンド-ペースト・バッファにカットします。                       |
| GOLD Remove     | 選択した領域をカット-アンド-ペースト・バッファにコピーします。                       |
| GOLD 上矢印        | フォント・グリフを 1 行上にずらします。                                  |
| GOLD 下矢印        | フォント・グリフを 1 行下にずらします。                                  |
| GOLD 左矢印        | フォント・グリフを 1 カラム左にずらします。                                |
| GOLD 右矢印        | フォント・グリフを 1 カラム右にずらします。                                |

<sup>a</sup>PF1 キーは、GOLD 状態を切り替えます。

以下に、一般的な `cedit` 操作を実行するために推奨する方法をまとめます。通常、1 つの編集操作を実行する方法は複数あります。

- グリフの描画

KP1 から KP9 までのキーを使用して、編集ウィンドウ内を移動します。これらのキーは、カーソル移動にバインドされています。キーは、KP5 キーを除き、コンパス上の点と見なすことができます。個々の点は、描画を行う向きを表します。描画は、KP3 キーで制御されるカーソル・モードに影響されます。カーソル・モードが移動モードに設定されている場合、描画キーは何も描画せずにカーソルを移動します。

KP5 キー (コンパスの真中のキー) を使用して、ピクセル状態をオンまたはオフに切り替えます。

カーソル移動は、それぞれ KP- および KP. キーにバインドされているタイプとラップ・モードの影響を受けます。

- グリフの編集

描画キーは、ピクセルを1つずつ変更します。複数の操作(カット、ペースト、およびコピー)は、ピクセルのブロックに影響します。まず選択機能により、選択領域を指定します。次にカットまたはコピー機能により、ピクセルのブロックをペースト・バッファに移動します。その後、カーソルを他の位置に移動し、ペースト機能を使用して、ペースト・バッファ内のピクセルを新しい位置に移動できます。ペースト操作は、ペースト・モードの設定に影響されます。

グリフ全体を特定の向きに移動するには、GOLD キーまたは PF1 キーと、該当する矢印キーを押します。

最後の編集操作を取り消すには、U キーを押します。

- 実寸大でのグリフの表示

端末エミュレーション・ウィンドウではなく、アジア系言語用の端末を使用している場合は、KP キーを押して、実寸大のグリフを表示できます。この操作は、デスクトップ・ウィンドウ環境ではサポートされません。

- グリフの複数プロトタイプの作成

複数のバージョンのグリフを作成して参照ウィンドウに格納しておけば、後で気に入ったものを選択できます。KP キーを押して、グリフを編集ウィンドウから参照ウィンドウに移動します。3つの参照ウィンドウは、左から右にラウンドロビン方式で使用されます。

この時点で、ポップアップ・メニューの Refer 機能を使用して、既存のグリフを、現在のデータベースまたは基準データベースから参照ウィンドウに移動できます。

- 編集ウィンドウの内のグリフと他のグリフとの置き換え

Use 機能は、グリフを編集ウィンドウに移動します。キーボードにバインドされている Use 機能は、現在のデータベースまたは基準データベース内の他のコードポイントからグリフをコピーします。ポップアップ・メニューからアクセスする Use 機能は、参照ウィンドウの1つから編集ウィンドウにグリフを移動します。

Use 機能は、編集ウィンドウ内の現在のグリフのコピーを使用バッファに保存します。このバッファからグリフを取り出すには、KP0 キーを押します。取り消しバッファの内容とは異なり、使用バッファ内のグリフは、編集操作中いつでも使用できます。

- 複数サイズのグリフの作成

cedit メイン・メニューの Scale オプションは、現在選択されているサイズを持つ、データベース内のすべてのグリフについて複数のサイズを作成します。フォント編集画面の Scale オプションは、現在編集中的文字だけに複数のサイズを作成します。既存の UDC データベースを使用している場合は、cedit のメイン・メニューではなく、フォント編集画面の Scale オプションを使用します。スケーリングが cedit のメイン・メニューから行われ、データベース全体に影響を及ぼす場合、スケーリングの後でフォントに対して手作業で加えた微調整は、すべてこの操作によって取り消されます。

- フォント編集画面の終了

Enter キーを押して編集結果を保存し、フォント編集画面を終了します。

編集結果を保存せずに終了するには、GOLD キーを押すか、あるいは PF2 キーと Enter キーを押します。

フォント・グリフを作成したら、その名前、入力キー・シーケンス、照合値、さらにオプションとして、文字が属するクラス名を指定する必要があります。cedit ユーザ・インタフェース画面の [Edit] メニュー項目を使用して、これらの属性を指定します。

## B.2 システム・ソフトウェアが使用する UDC サポート・ファイルの作成

UDC データベースに格納されている文字属性は、さまざまなシステム・ソフトウェアのニーズに合わせて、特定の種類のファイルに出力されなければなりません。たとえば、端末ドライバ・ソフトウェアと asort コマンドは、ユーザ定義文字属性を認識できなければなりませんが、UDC データベース内の情報に直接アクセスすることはできません。そのため、UDC データベース内の文字属性を作成したり変更したときは、cgen コマンドを使用して、次のサポート・ファイルを作成する必要があります。

- SoftODL (ソフトウェア・オンデマンド・ローディング) サービスが使用するフォント・ファイル
- デバイスに直接ロードできるフォント・ファイル
- 文字をソートするための照合値テーブル
- UDC の入力キー・シーケンスを含むファイル
- X および Motif アプリケーションが使用するフォント・ファイル

次のコマンドは、`~wang/.udc` 内の UDC データベースについて、これらのファイルのいくつかを作成します。

```
% cgen -odl -pre -col -iks ~wang/.udc
```

オプションを指定せずに `cgen` を実行すると、指定したデータベースに関する統計情報が表示されます。UDC データベースを指定せずに実行すると、一般ユーザの場合は個人用のユーザ・データベースが使用されます。スーパーユーザの場合には、システム・データベースが使用されます。つまり、上記の例で、コマンドを入力したユーザが `wang` でログオンしている場合、データベースの指定は必要ありません。

表 B-9 で、`cgen` コマンドのオプションについて説明します。この表で、`bdf` フォーマットは Bitmap Distribution Format のことで、`pcf` フォーマットは Portable Compiled Format のことです。これらのフォーマットについては、`bdf`to`pcf`(1X) のリファレンス・ページを参照してください。

表 B-9: `cgen` コマンドのオプション

| オプション                               | 説明                                                                                                                          |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>-bdf</code>                   | X および DECwindows の Motif アプリケーションに必要な <code>.bdf</code> (Bitmap Distribution Format) ファイルを作成します。                            |
| <code>-col</code>                   | 照合値テーブルを作成します。これらのテーブルをソート操作に使用したいときは、 <code>sort</code> コマンドではなく <code>asort</code> コマンドを使用しなければなりません。                     |
| <code>-dpi 75 100</code>            | <code>-bdf</code> と <code>-pcf</code> オプションを指定して、 <code>.bdf</code> と <code>.pcf</code> ファイルを作成するときの解像度を 75 または 100 に設定します。 |
| <code>-fprop <i>property</i></code> | <code>-bdf</code> と <code>-pcf</code> オプションを指定して、 <code>.bdf</code> と <code>.pcf</code> ファイルを作成するときのフォント・プロパティを設定します。       |
| <code>-iks</code>                   | 入力キー・シーケンス・ファイルを作成します。                                                                                                      |

表 B-9: cgen コマンドのオプション (続き)

| オプション                      | 説明                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -merge <i>font_pattern</i> | fontconverter コマンドを呼び出して、指定された <i>font_pattern</i> (たとえば、'*-140-*jisx0208*') に合った既存の .pcf フォント・ファイルと UDC フォントをマージします。<br><br>-merge オプションを指定するときは、-pcf と -size オプションも指定する必要があります。出力の .pcf ファイルは、 <i>registry_width_height.pcf</i> の形式です。registry は、指定されたフォント・ファイルのフォント・レジストリ・フィールドです。                                                                                                                                    |
| -osiz <i>width-xheight</i> | bdf 出力フォーマットのフォント・サイズを指定します。<br><br>bdf フォーマットのフォント・サイズは、UDC データベースで定義されているフォント・サイズと異なる場合があります。cedit コマンドがサポートするフォント・サイズには制限があります。-osiz オプションを指定すると、.bdf ファイルと、.bdf ファイルから生成される .pcf ファイルの両方で、これらのサイズ制限を変更できます。<br><br>-osiz オプションで指定したサイズ・パラメータが、-size オプションで指定したサイズ・パラメータよりも小さい場合は、UDC フォント・グリフの左上部分だけが使用されます。-osiz オプションで指定したサイズ・パラメータが、-size オプションで指定したサイズ・パラメータよりも大きい場合には、結果として得られたフォント・グリフの右下部分が OFF ピクセルで塗り潰されます。 |
| -pcf                       | bdf2topcf コマンドを呼び出して、X と Motif アプリケーションに必要な .pcf ファイルを作成します。<br><br>このオプションを指定した場合、cgen コマンドは mkfontdir と xset コマンドも呼び出して、フォントをフォント・サーバに通知し、アプリケーションから利用できるようにします。                                                                                                                                                                                                                                                       |



表 B-9: cgen コマンドのオプション (続き)

| オプション                | 説明                                                                                                                                                                                                                                                                             |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -pre                 | <p>プリロード・フォント・ファイルを作成します。</p> <p>プリロード・フォント・ファイルは、端末と一部のプリンタに直接かつ完全にロードされるファイルです。ほとんどのデバイスではメモリに制限があるため、UDC データベースが大きい場合には、プリロード・ファイルを使用しても意味がありません。オンデマンド・ローディング (ODL) では、プリロード・フォント・ファイルの代わりに ODL フォント・ファイルを使用します。</p>                                                       |
| -odl                 | <p>ODL フォント・ファイルを作成します。</p> <p>端末ドライバは、必要性和使用可能なメモリに応じて、ODL フォント・ファイルからフォントをインクリメンタルにロードします。</p>                                                                                                                                                                               |
| -win <i>userfont</i> | <p>Windows Version 3.1 や Windows NT Version 3.5 システムにコピー可能な、<i>userfont</i> というフォント・ファイルを作成します。指定したファイルには1つのサイズしか使用できないため、-size フラグを指定する必要があります。このオプションで作成されるフォント・ファイルでサポートされるコードセットは、big5 (中国語 Windows システム)、SJIS (日本語 Windows システム)、および deckorean (韓国語 Windows システム) です。</p> |

### B.3 X11 や Motif アプリケーションで使用する UDC フォントの処理

cgen の -pre オプションで作成されたプリロード・フォント・ファイルを、X11 や Motif アプリケーションで使えるようにするには、BDF (Bitmap Distribution Format) または PCF (Portable Compiled Format) に変換しなければなりません。この変換を行う fontconverter コマンドは、変換出力に対し次のいずれかの処理を実行できます。

- 独立した pcf および bdf フォント・ファイルを作成する。この場合は、これらのファイルをワークステーションにインストールして、アプリケーションから使えるようにしなければなりません。
- フォントを既存の (pcf) フォント・ファイルにマージする。

ここでは、fontconverter コマンドと、そのオプションの使用方法について説明します。cgen コマンドにも fontconverter のオプションに対応するオプションがあります。つまり、fontconverter の操作は、cgen のコマンド行で類似したオプションを指定することにより、間接的に実行できます。

### B.3.1 fontconverter のコマンド・オプションの使用方法

次の例は、出力ファイルに省略時の名前を使用する最も簡単な形式の fontconverter コマンドです。この例とこれ以降の説明では、コマンド入力時のロケールには日本語ロケールが設定されており、24x24 のフォント・サイズがフォント・グリフの作成時に cedit エディタで指定されているものとします。

```
% fontconverter \  
-font -jdecw-screen-medium-r-normal--24-240-75-75-m-240-jisx0208-kanji11 \  
my_font.pre
```

上記のコマンドは、my\_fonts.pre ファイル内のフォントを変換します。省略時の設定では、コマンドはフォント・ファイル JISX.UDC\_24\_24.pcf と JISX.UDC\_24\_24.bdf を作成します。

出力フォント・ファイルの省略時のベース名は、次のように言語によって異なります。

- 日本語: JISX.UDC
- 中国語 (繁体字): DEC.CNS.UDC
- 中国語 (簡体字): GB.UDC

フォントの幅と高さを表す文字列が、出力フォント・ファイル名のベース名に自動的に追加されます。ベース名は、レジストリ名として、XLFD (X Logical Font Description) でも使用されます。フォントをアプリケーションから利用できるようにするためには、コンパイル済みの (pcf) フォントに対して、次のいずれかの処理を行う必要があります。

- フォントが置かれているディレクトリで、次のコマンドを実行する。

```
% /usr/bin/X11/mkfontdir  
% /usr/bin/X11/xset +fp `pwd`  
% /usr/bin/X11/xset fp rehash
```

これらのコマンドは、サーバが再起動するか、システムがシャットダウンするまで、フォントを使用できるようにします。

cgen コマンドで `-pcf` オプションを指定すると、`fontconverter` コマンドと `mkfontdir` コマンドを実行できます。

- フォントを永続的に (つまり、サーバの再起動やシステムのシャットダウンの後でも) 使用できるようにするには、次のコマンドを使用する。
  1. `/usr/i18n/usr/lib/X11/fonts/decwin/100dpi` などの既存のフォント・ディレクトリに `pcf` フォントをコピーします。

```
% cp JISX.UDC_24_24.pcf \
/usr/i18n/usr/lib/X11/fonts/decwin/100dpi
```
  2. そのディレクトリに移動します。

```
% cd /usr/i18n/usr/lib/X11/fonts/decwin/100dpi
```
  3. そのディレクトリで `mkfontdir` コマンドを入力します。

```
% /usr/bin/X11/mkfontdir
```
  4. 次の `xset` コマンドを入力します。

```
% /usr/bin/X11/xset fp rehash
```

表 B-10 では、`fontconverter` コマンドのオプションを説明します。オプションは、`-preload` を除き、コマンド行に指定する順に示されています。これらのオプションの使用例については、B.3.2 項を参照してください。

表 B-10: `fontconverter` コマンドのオプションと引数

| 引数またはオプション          | 説明                                                                                            |
|---------------------|-----------------------------------------------------------------------------------------------|
| <code>-merge</code> | コマンド出力を、既存のフォント・ファイルとマージします。<br><code>-font</code> オプションも参照してください。                            |
| <code>-w</code>     | フォント幅を指定します。<br>このオプションは、フォントが <code>cedit</code> フォント編集ウィンドウで指定された幅よりも狭い幅で作成される場合に使用します。     |
| <code>-h</code>     | フォントの高さを指定します。<br>このオプションは、フォントが <code>cedit</code> フォント編集ウィンドウで指定された高さよりも低い高さで作成される場合に使用します。 |

表 B-10: fontconverter コマンドのオプションと引数 (続き)

| 引数またはオプション                   | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -udc <i>base_name</i>        | 出力 UDC フォント・ファイルの基本ファイル名を指定します。<br><br>このオプションは、スタンドアロンの出力ファイルを作成し (出力を既存のファイルにマージしない)、出力ファイルに省略時の名前を付けたくない場合に使用します。                                                                                                                                                                                                                                                                                                                                       |
| -font <i>reference_font</i>  | 基準フォントを指定します。基準フォントは、使用しているディスプレイ上で使用可能なフォントの名前です。xlsfonts コマンド (xlsfonts(1X) を参照) を使用して、使用可能なフォントを判別します。<br><br>-font オプションを -merge オプションと組み合わせて使用すると、 <i>reference_font</i> は、変換されたフォント・グリフがマージされるフォントを示します。<br><br>-font オプションを -merge オプションなしで使用すると、 <i>reference_font</i> のヘッダが、スタンドアロン出力ファイルのヘッダを生成するときの基準として使用されます。また、 <i>reference_font</i> の情報は、スタンドアロン出力ファイル内の省略時の文字を決定するのにも使用されます。省略時の文字は、フォントが指定されたコードに対応するグリフをなにも含まないときに表示されるグリフ (通常は四角形) です。 |
| -preload <i>preload_font</i> | 入力ファイル (cgen-pre コマンドで作成されたもの) を指定します。<br><br>このオプションは、fontconverter コマンド行の任意の位置に <i>preload_font</i> 引数を指定したい場合に使用します。 <i>preload_font</i> をコマンド行の終わりに置くときは、-preload を省略できます。                                                                                                                                                                                                                                                                             |

B.3.2 出力ファイル・フォーマットの制御

X と Motif アプリケーションには、PCF フォーマットのロード可能なフォントが必要です。

-merge オプションを使用しない場合、fontconverter コマンドは、PCF と BDF の両方のフォーマットのスタンドアロン・フォント・ファイルを作成します。-merge オプションを指定すると、変換されたフォントは、-font オプションで指定された標準 PCF フォントにマージされ、PCF フォーマットのスタンドアロン・ファイルのみが作成されます。

UDC フォントを標準のフォントとマージしたときは、すべての Motif アプリケーションで結合したファイルを使用できます。

独立したフォント・ファイルを作成した場合、マージしたフォントは、結合したファイルを明示的にロードするアプリケーションで使用できます。フォント・レジストリが、特定のロケールの UDC レジストリの 1 つであれば、標準のシステム・アプリケーションでも作成したフォント・ファイルを使用できます。

fontconverter の処理時間は、フォントを既存のファイルにマージするときの方が、独立したファイルを作成するときよりも長くなる点に注意してください。

次のような操作を実行する fontconverter の例を、以下に示します。

- ファイル `udc_font.pre` 内のプリロード・フォーマット・フォントを PCF フォーマットに変換する。
- 変換された出力をフォント `-jdecw-screen-medium-r-normal--24-240-75-75-m-240-jisx0208-kanji11` にマージする。
- 標準のフォントと新しいフォントを組み合わせ、出力ファイル `JISX0208-Kanji11_24_24.pcf` を生成する。

```
% fontconverter -merge -font \  
-jdecw-screen-medium-r-normal--24-240-75-75-m-240-jisx0208-kanji11 \  
udc_font.pre
```

次のような操作を実行する fontconverter の例を、以下に示します。

- ファイル `deckanji.udc_24_24.bdf` と `deckanji.udc_24_24.pcf` を作成する。
- フォント `-jdecw-screen-medium-r-normal--24-24-240-75-75-m-240-jisx0208-kanji11` からこれらのファイルの省略時の文字と大部分のヘッダ情報を取得する。
- フォント・レジストリ・フィールドを `deckanji.udc` に設定する。

```
% fontconverter -udc deckanji.udc -font \  
-jdecw-screen-medium-r-normal--24-240-75-75-m-240-jisx0208-kanji11 \  
udc_font.pre
```

# C

## プログラムでの DECterm ローライゼーション機能の使用

ここでは、DECterm 端末エミュレータで用意されている、ローカル言語サポートのためのプログラミング機能について説明します。

### C.1 DECterm ウィンドウにおける罫線の処理

ビデオ端末用のプログラミング・マニュアルには、ANSI エスケープ・シーケンスを使用して、文字の挿入と削除、空白行の挿入と削除、倍の高さと幅を持つ文字の表示などの操作をどのように行うかが説明されています。DECterm ウィンドウはエミュレータであるため、これらのエスケープ・シーケンスは、DECterm ウィンドウにテキストやグラフィックスを表示するプログラムにも使用できます。

オペレーティング・システムが備えるアジア系言語のための拡張機能には、DECterm ウィンドウ内の指定した領域に罫線を引いたり削除するためのエスケープ・シーケンスが追加されています。これらのエスケープ・シーケンスを使用することにより、アプリケーションで表やダイアグラムを描画できます。

以降の節では、パターンと領域パラメータに従って罫線を引いたり削除するためのエスケープ・シーケンスについて説明します。

#### C.1.1 パターンに従った罫線の描画

ニーモニック DECDRLBR で表されるエスケープ・シーケンスは、指定されたパターンに従って四角形の境界上に罫線を引きます。DECDRLBR のフォーマットは、次のとおりです。

```
CSI P1;Px;Plx;Py;Ply ,r
```

意味は、次のとおりです。

- *P1* は、罫線の描画パターンを示します。

$P1$  は、四角形のすべての辺に罫線を引く、左右の辺だけに罫線を引く、上下の辺だけに罫線を引くなどのパターンを示します。

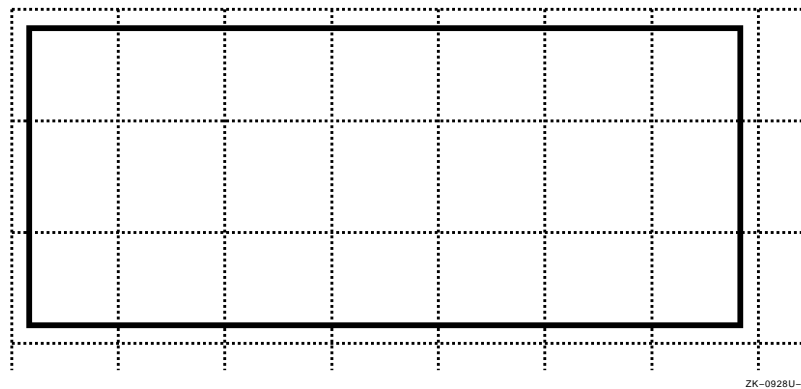
- $Px$  は、開始点の絶対位置をカラムで示します。
- $Plx$  は、領域の幅をカラム数で示します。
- $Py$  は、開始点の絶対位置を行で示します。
- $Ply$  は、領域の高さを行数で示します。

DECterm ソフトウェアは、アプリケーションから DECRLBR エスケープ・シーケンスを受け取ると、 $P1$  に指定されたパターンに従って、座標  $(Px, Py)$  と  $(Px + Plx - 1, Py + Ply - 1)$  間の領域の 1 つまたは複数の境界上に罫線を引きます。次の例を考えてみます。

```
CSI 15 ; 1 ; 5 ; 1 ; 2 , r
```

上記のエスケープ・シーケンスにより、DECterm ソフトウェアは図 C-1 に示す罫線を引きます。

図 C-1: DECRLBR シーケンスを用いた罫線の描画

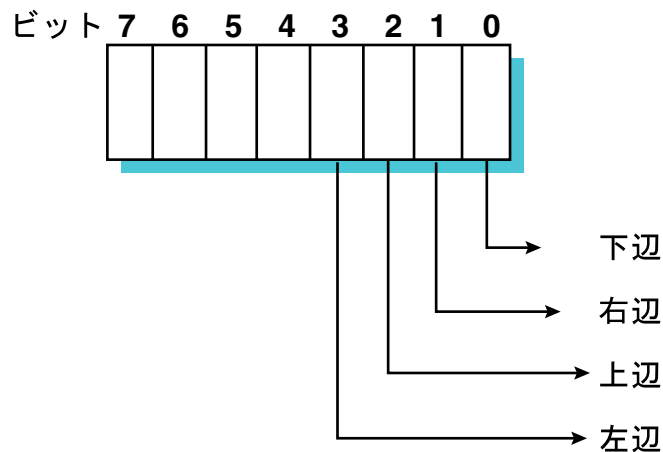


DECterm ソフトウェアは、1 ピクセル幅の罫線を描画できます。画面をスクロールすると、罫線はテキストと同様にスクロールします。

図 C-2 とその後の表は、DECRLBR パラメータに対応するビット・パターンを示します。



図 C-2: DECDRLBR パラメータのビット・パターン



ZK-0931U-AIJ

| ビット   | ビット値 | 説明          |
|-------|------|-------------|
| ビット 0 | 1    | 下辺に罫線を引きます。 |
| ビット 1 | 2    | 右辺に罫線を引きます。 |
| ビット 2 | 4    | 上辺に罫線を引きます。 |
| ビット 3 | 8    | 左辺に罫線を引きます。 |

次のリストで、DECDRLBR パラメータについてより詳細に説明します。

- 罫線のパターン ( $P1$ )

パターンは、領域境界上にどのように罫線を引くかを制御するビットマスクです。罫線は、境界線上のビットのオン/オフによって描画されます。たとえば、 $P1$  が 15 に設定されていれば、すべての境界上に罫線が引かれ、 $P1$  が 5 に設定されていれば、上下の境界上に罫線が引かれます。例を次に示します。

|      |         |         |         |         |      |
|------|---------|---------|---------|---------|------|
| 境界 : | 下辺      | 右辺      | 上辺      | 左辺      |      |
| $P1$ | = ビット 0 | + ビット 1 | + ビット 2 | + ビット 3 |      |
| $P1$ | = 1     | + 2     | + 4     | + 8     | = 15 |
| $P1$ | = 1     | + 4     |         |         | = 5  |

- 開始点 ( $Px, Py$ ) の絶対位置

$Px$  は開始カラム位置であり、 $Py$  は開始行位置です。これらのパラメータを省略するか、明示的に 0 (ゼロ) に設定したときは、開始位置はカラム 1 と行 1 になります。つまり、四角形の左上隅が座標 (1,1) になります。

- 領域のサイズ ( $Plx, Ply$ )

$Plx$  はカラム数による領域の幅であり,  $Ply$  は行数による領域の高さです。これらのパラメータを省略するか, 明示的に 0 (ゼロ) に設定したときは, 領域は 1 カラム幅で 1 行の高さになります。

### C.1.2 パターンによる罫線の削除

エスケープ・シーケンス DECERLBRP は, 指定されたパターンに従って四角形の境界上の罫線を削除します。DECERLBRP のフォーマットは, 次のとおりです。

CSI  $P1; Px; Plx; Py; Ply, s$

意味は次のとおりです。

- $P1$  は, 罫線の描画パターンを示します。  
 $P1$  は, 四角形のすべての辺に罫線を引く, 左右の辺だけに罫線を引く, 上下の辺だけに罫線を引くなどのパターンを示します。
- $Px$  は, 開始点の絶対位置をカラムで示します。
- $Plx$  は, 領域の幅をカラム数で示します。
- $Py$  は, 開始点の絶対位置を行で示します。
- $Ply$  は, 領域の高さを行数で示します。

### C.1.3 領域内のすべての罫線の削除

エスケープ・シーケンス DECERLBRA は, 領域境界上に引かれた罫線だけでなく, 矩形領域内のすべての罫線を削除します。DECERLBRA のフォーマットは, 次のとおりです。

CSI  $P1; Px; Plx; Py; Ply, t$

意味は次のとおりです。

- $P1$  は, 領域が画面全体なのか, 画面の一部なのかを指定します。  
 $P1$  の値が 1 の場合, DECterm ソフトウェアは画面上のすべての罫線を削除します。その場合, パラメータ  $Px, Plx, Py$ , および  $Ply$  は無視されます。 $P1$  の値が 2 の場合, DECterm ソフトウェアは, パラメータ  $Px, Plx, Py$ , および  $Ply$  で定義される矩形領域内のすべての罫線を削除します。 $P1$  が省略されているか, 明示的に 0 (ゼロ) に設定されてい

る場合、DECterm ソフトウェアは画面上のすべての罫線を削除します (P1 の値が、省略時の値 1 の場合と同じ結果)。

- Px は、開始点の絶対位置をカラムで示します。
- Plx は、領域の幅をカラム数で示します。
- Py は、開始点の絶対位置を行で示します。
- Ply 領域の高さを行数で示します。

C.1.4 その他の DECterm エスケープ・シーケンス

表 C-1 に、画面上に罫線を引く際の標準 DECterm エスケープ・シーケンスの効果を示します。

表 C-1: 罫線に対する標準エスケープ・シーケンスの効果

| ニーモニック | 説明 | 罫線への効果 |
|--------|----|--------|
|--------|----|--------|

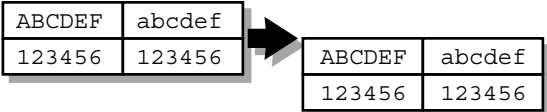
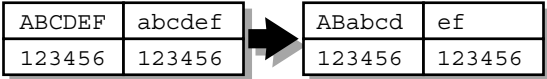
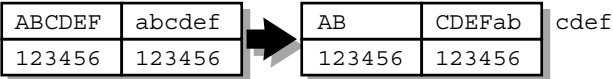

|                                  |                         |                                                                                                                                                           |
|----------------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| DECDWL ,<br>DECDHLT ,<br>DECDHLB | 2 倍の幅、または 2 倍の高さで表示します。 | これらのエスケープ・シーケンスは、幅が常に 1 ピクセルの罫線には影響しません。また、エスケープ・シーケンスが、テキストの高さと幅を 2 倍にするパラメータを指定して使用された場合でも、罫線の表示を制御するエスケープ・シーケンスのパラメータ単位は、カラムの幅と行の高さが常に等間隔になるように設定されます。 |
|----------------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|

|     |                    |                                                                                                    |
|-----|--------------------|----------------------------------------------------------------------------------------------------|
| GSM | グラフィックス・サイズを変更します。 | これらのエスケープ・シーケンスは、幅が常に 1 ピクセルの罫線には影響しません。DECDWL , DECDHLT , および DECDHLB に対する上記のコメントは、GSM にも当てはまります。 |
|-----|--------------------|----------------------------------------------------------------------------------------------------|

|                  |                                   |                                                        |
|------------------|-----------------------------------|--------------------------------------------------------|
| ED , EL ,<br>ECH | 画面の削除、<br>行の削除、および文字の削除<br>実行します。 | これらのエスケープ・シーケンスは罫線の削除は行わず、罫線の内側にある文字だけを削除します。次に例を示します。 |
|------------------|-----------------------------------|--------------------------------------------------------|

|    |              |                                                                                          |
|----|--------------|------------------------------------------------------------------------------------------|
| DL | 行を削除し<br>ます。 | このエスケープ・シーケンスは、削除操作を行っている行の文字と罫線の両方を削除します。削除位置の後のテキスト行とそれに対応する罫線が上方向にスクロールされます。次に例を示します。 |
|----|--------------|------------------------------------------------------------------------------------------|

表 C-1: 罫線に対する標準エスケープ・シーケンスの効果 (続き)

| 二一モニック          | 説明                                  | 罫線への効果                                                                                                                                                                    |
|-----------------|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IL              | 行を挿入します。                            | このエスケープ・シーケンスは、現在の位置に空白行を挿入します。現在の位置にあるテキスト行とそれに対応する罫線が下方方向にスクロールされます。次に例を示します。<br><br> |
| DCH             | 文字を削除します。                           | このエスケープ・シーケンスは、罫線を削除しません。次の例は、第3カラムで4文字を削除します。<br><br>                                  |
| ICH             | 文字を挿入します。                           | このエスケープ・シーケンスは現在の位置に空白を挿入しますが、罫線には影響しません。次の例は、第3カラムに4文字を挿入します。<br><br>                |
| IRM             | 挿入/置換モードに入ります。                      | 挿入/置換モードは、罫線には影響しません。次の例は、第3カラムに文字 w, x, y, z を挿入し、文字 f を s に置換します。<br><br>           |
| DECCOLM         | カラム・モードに入ります。                       | カラム・モードが有効なときは、テキストとそれに対応する罫線が削除されます。                                                                                                                                     |
| RIS ,<br>DECSTR | 初期状態とソフト端末をリセットし、リセットSETUPモードに入ります。 | RIS シーケンスは画面上のすべての罫線を削除しますが、DECSTR シーケンスは罫線を削除しません。DECterm の Commands メニューの Clear Display オプションは、すべての罫線を削除しますが、Reset Terminal オプションは罫線を削除しません。                            |

### C.1.5 DECterm が罫線をサポートしているかどうかの判定

アプリケーションで罫線を描画できる機能は、DECterm ウィンドウがこの機能をサポートする端末タイプをエミュレートしている場合にのみ使用できます。アプリケーションは、DECterm ソフトウェアから 1 次デバイス属性を取得することにより、デバイスがサポートされているかどうかを検査できます。

VT 端末と DECterm ソフトウェアは、アプリケーションからの要求に応じて 1 次デバイス属性をレポートします。レポートに拡張値 43 が含まれているときは、デバイスで罫線を描画できます。この機能は、level-2 以降のビデオ・ディスプレイで使用できます。たとえば、DECterm ウィンドウが VT382 の日本語版、VT382-J 端末をエミュレートしている場合、1 次デバイス属性は、次のようにレポートされます。

```
CSI ? 63 ; 1 ; 2 ; 4 ; 5 ; 6 ; 7 ; 8 ; 10 ; 15 ; 43 c
```

アプリケーションは CSI c または CSI 0 c のいずれかのエスケープ・シーケンスを VT 端末、あるいは DECterm ソフトウェアに送ることにより、デバイス属性レポートを取得できます。

## C.2 DECterm プログラミングにおける制約

この節では、ハードウェア・マニュアルに記載されている端末プログラミング機能に関する DECterm ソフトウェアの制約について説明します。

### C.2.1 ダウンライン・ローダブル文字

DECterm ソフトウェアは、端末のプリロードとオンデマンド・ローディングに使用されるダウンライン・ローダブル文字をサポートしません。DECterm ソフトウェアは、これらの文字のエスケープ・シーケンスを無視します。

### C.2.2 DRCS 文字

DECterm ソフトウェアは、DRCS (DIGITAL Replacement Character Set) の標準文字セット (SCS) コンポーネント以外はサポートしません。DECterm ソフトウェアは SCS 文字を受け取ると、X ウィンドウ・サーバの `--dec-drcs` という XLFD 名を持つフォントを検索し、ソフト文字セットとして処理します。また、DECterm ソフトウェアは、端末プログラミング・アプリケーションから送られた DECDLD 制御文字列を無視します。



# D

## サンプル・ロケールのソース・ファイル

この付録では、第 6 章で説明しているサンプル・ロケールの全ソース・ファイルを示します。

### D.1 文字マップ (charmap) のソース・ファイル

この節のサンプルは、fr\_FR.ISO8859-1@example ロケール用の ISO8859-1.cmap ファイルです。

```
#
#   Charmap for ISO 8859-1 codeset
#
#

<code_set_name>                "ISO8859-1"
<mb_cur_max>                   1
<mb_cur_min>                   1
<escape_char>                  \
<comment_char>                 #

CHARMAP

#   Portable characters and other standard
#   control characters

<NUL>                          \x00
<SOH>                          \x01
<STX>                          \x02
<ETX>                          \x03
<EOT>                          \x04
<ENQ>                          \x05
<ACK>                          \x06
<BEL>                          \x07
<alert>                        \x07
<backspace>                    \x08
<tab>                          \x09
<newline>                      \x0a
<vertical-tab>                 \x0b
<form-feed>                    \x0c
<carriage-return>             \x0d
<SO>                          \x0e
```

|                     |      |
|---------------------|------|
| <SI>                | \x0f |
| <DLE>               | \x10 |
| <DC1>               | \x11 |
| <DC2>               | \x12 |
| <DC3>               | \x13 |
| <DC4>               | \x14 |
| <NAK>               | \x15 |
| <SYN>               | \x16 |
| <ETB>               | \x17 |
| <CAN>               | \x18 |
| <EM>                | \x19 |
| <SUB>               | \x1a |
| <ESC>               | \x1b |
| <IS4>               | \x1c |
| <IS3>               | \x1d |
| <IS2>               | \x1e |
| <IS1>               | \x1f |
| <SP>                | \x20 |
| <space>             | \x20 |
| <exclamation-mark>  | \x21 |
| <quotation-mark>    | \x22 |
| <number-sign>       | \x23 |
| <dollar-sign>       | \x24 |
| <percent-sign>      | \x25 |
| <ampersand>         | \x26 |
| <apostrophe>        | \x27 |
| <left-parenthesis>  | \x28 |
| <right-parenthesis> | \x29 |
| <asterisk>          | \x2a |
| <plus-sign>         | \x2b |
| <comma>             | \x2c |
| <hyphen>            | \x2d |
| <hyphen-minus>      | \x2d |
| <period>            | \x2e |
| <full-stop>         | \x2e |
| <slash>             | \x2f |
| <solidus>           | \x2f |
| <zero>              | \x30 |
| <one>               | \x31 |
| <two>               | \x32 |
| <three>             | \x33 |
| <four>              | \x34 |
| <five>              | \x35 |
| <six>               | \x36 |
| <seven>             | \x37 |
| <eight>             | \x38 |
| <nine>              | \x39 |
| <colon>             | \x3a |
| <semicolon>         | \x3b |
| <less-than-sign>    | \x3c |



|                        |      |
|------------------------|------|
| <equals-sign>          | \x3d |
| <greater-than-sign>    | \x3e |
| <question-mark>        | \x3f |
| <commercial-at>        | \x40 |
| <A>                    | \x41 |
| <B>                    | \x42 |
| <C>                    | \x43 |
| <D>                    | \x44 |
| <E>                    | \x45 |
| <F>                    | \x46 |
| <G>                    | \x47 |
| <H>                    | \x48 |
| <I>                    | \x49 |
| <J>                    | \x4a |
| <K>                    | \x4b |
| <L>                    | \x4c |
| <M>                    | \x4d |
| <N>                    | \x4e |
| <O>                    | \x4f |
| <P>                    | \x50 |
| <Q>                    | \x51 |
| <R>                    | \x52 |
| <S>                    | \x53 |
| <T>                    | \x54 |
| <U>                    | \x55 |
| <V>                    | \x56 |
| <W>                    | \x57 |
| <X>                    | \x58 |
| <Y>                    | \x59 |
| <Z>                    | \x5a |
| <left-square-bracket>  | \x5b |
| <backslash>            | \x5c |
| <reverse-solidus>      | \x5c |
| <right-square-bracket> | \x5d |
| <circumflex>           | \x5e |
| <circumflex-accent>    | \x5e |
| <underscore>           | \x5f |
| <low-line>             | \x5f |
| <grave-accent>         | \x60 |
| <a>                    | \x61 |
| <b>                    | \x62 |
| <c>                    | \x63 |
| <d>                    | \x64 |
| <e>                    | \x65 |
| <f>                    | \x66 |
| <g>                    | \x67 |
| <h>                    | \x68 |
| <i>                    | \x69 |
| <j>                    | \x6a |
| <k>                    | \x6b |

|                       |      |
|-----------------------|------|
| <l>                   | \x6c |
| <m>                   | \x6d |
| <n>                   | \x6e |
| <o>                   | \x6f |
| <p>                   | \x70 |
| <q>                   | \x71 |
| <r>                   | \x72 |
| <s>                   | \x73 |
| <t>                   | \x74 |
| <u>                   | \x75 |
| <v>                   | \x76 |
| <w>                   | \x77 |
| <x>                   | \x78 |
| <y>                   | \x79 |
| <z>                   | \x7a |
| <left-brace>          | \x7b |
| <left-curly-bracket>  | \x7b |
| <vertical-line>       | \x7c |
| <right-brace>         | \x7d |
| <right-curly-bracket> | \x7d |
| <tilde>               | \x7e |
| <DEL>                 | \x7f |

```
#
# Extended control characters
# (names taken from ISO 6429)
#
```

|       |      |
|-------|------|
| <PAD> | \x80 |
| <HOP> | \x81 |
| <BPH> | \x82 |
| <NBH> | \x83 |
| <IND> | \x84 |
| <NEL> | \x85 |
| <SSA> | \x86 |
| <ESA> | \x87 |
| <HTS> | \x88 |
| <HTJ> | \x89 |
| <VTS> | \x8a |
| <PLD> | \x8b |
| <PLU> | \x8c |
| <RI>  | \x8d |
| <SS2> | \x8e |
| <SS3> | \x8f |
| <DCS> | \x90 |
| <PU1> | \x91 |
| <PU2> | \x92 |
| <STS> | \x93 |
| <CCH> | \x94 |
| <MW>  | \x95 |

|        |      |
|--------|------|
| <SPA>  | \x96 |
| <EPA>  | \x97 |
| <SOS>  | \x98 |
| <SGCI> | \x99 |
| <SCI>  | \x9a |
| <CSI>  | \x9b |
| <ST>   | \x9c |
| <OSC>  | \x9d |
| <PM>   | \x9e |
| <APC>  | \x9f |

```
#
# Other graphic characters
#
```

|                             |      |
|-----------------------------|------|
| <nobreakspace>              | \xa0 |
| <inverted-exclamation-mark> | \xa1 |
| <cent>                      | \xa2 |
| <sterling>                  | \xa3 |
| <pound>                     | \xa3 |
| <currency>                  | \xa4 |
| <yen>                       | \xa5 |
| <broken-bar>                | \xa6 |
| <section>                   | \xa7 |
| <diaeresis>                 | \xa8 |
| <diaeresis>                 | \xa8 |
| <copyright>                 | \xa9 |
| <feminine>                  | \xaa |
| <guillemot-left>            | \xab |
| <not>                       | \xac |
| <dash>                      | \xad |
| <registered>                | \xae |
| <macron>                    | \xaf |
| <degree>                    | \xb0 |
| <ring>                      | \xb0 |
| <plus-minus>                | \xb1 |
| <superscript-two>           | \xb2 |
| <superscript-three>         | \xb3 |
| <acute>                     | \xb4 |
| <mu>                        | \xb5 |
| <micro>                     | \xb5 |
| <paragraph>                 | \xb6 |
| <dot>                       | \xb7 |
| <cedilla>                   | \xb8 |
| <superscript-one>           | \xb9 |
| <masculine>                 | \xba |
| <guillemot-right>           | \xbb |
| <one-quarter>               | \xbc |
| <one-half>                  | \xbd |

|                          |      |
|--------------------------|------|
| <three-quarters>         | \xbe |
| <inverted-question-mark> | \xbf |
| <A-grave>                | \xc0 |
| <A-acute>                | \xc1 |
| <A-circumflex>           | \xc2 |
| <A-tilde>                | \xc3 |
| <A-diaeresis>            | \xc4 |
| <A-ring>                 | \xc5 |
| <AE-ligature>            | \xc6 |
| <C-cedilla>              | \xc7 |
| <E-grave>                | \xc8 |
| <E-acute>                | \xc9 |
| <E-circumflex>           | \xca |
| <E-diaeresis>            | \xcb |
| <I-grave>                | \xcc |
| <I-acute>                | \xcd |
| <I-circumflex>           | \xce |
| <I-diaeresis>            | \xcf |
| <ETH-icelandic>          | \xd0 |
| <N-tilde>                | \xd1 |
| <O-grave>                | \xd2 |
| <O-acute>                | \xd3 |
| <O-circumflex>           | \xd4 |
| <O-tilde>                | \xd5 |
| <O-diaeresis>            | \xd6 |
| <multiplication>         | \xd7 |
| <O-slash>                | \xd8 |
| <U-grave>                | \xd9 |
| <U-acute>                | \xda |
| <U-circumflex>           | \xdb |
| <U-diaeresis>            | \xdc |
| <Y-acute>                | \xdd |
| <THORN-icelandic>        | \xde |
| <s-sharp>                | \xdf |
| <a-grave>                | \xe0 |
| <a-acute>                | \xe1 |
| <a-circumflex>           | \xe2 |
| <a-tilde>                | \xe3 |
| <a-diaeresis>            | \xe4 |
| <a-ring>                 | \xe5 |
| <ae-ligature>            | \xe6 |
| <c-cedilla>              | \xe7 |
| <e-grave>                | \xe8 |
| <e-acute>                | \xe9 |
| <e-circumflex>           | \xea |
| <e-diaeresis>            | \xeb |
| <i-grave>                | \xec |
| <i-acute>                | \xed |
| <i-circumflex>           | \xee |
| <i-diaeresis>            | \xef |

```

<eth-icelandic>          \xf0
<n-tilde>                 \xf1
<o-grave>                 \xf2
<o-acute>                 \xf3
<o-circumflex>           \xf4
<o-tilde>                 \xf5
<o-diaeresis>            \xf6
<division>               \xf7
<o-slash>                 \xf8
<u-grave>                 \xf9
<u-acute>                 \xfa
<u-circumflex>           \xfb
<u-diaeresis>            \xfc
<y-acute>                 \xfd
<thorn-icelandic>        \xfe
<y-diaeresis>            \xff

```

END CHARMAP

## D.2 ロケール定義のソース・ファイル

この節のサンプルは、第 6 章 の例で使用されている `fr_FR.ISO8859-1@example.src` ファイルです。

```

#####
# Locale Source for fr_FR (French in France) locale          #
#####
LC_CTYPE
#####

upper  <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
        <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>;\
        <A-grave>;\
        <A-circumflex>;\
        <AE-ligature>;\
        <C-cedilla>;\
        <E-grave>;\
        <E-acute>;\
        <E-circumflex>;\
        <E-diaeresis>;\
        <I-circumflex>;\
        <I-diaeresis>;\
        <O-circumflex>;\
        <U-grave>;\
        <U-circumflex>;\
        <U-diaeresis>

lower  <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
        <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>;\
        <a-grave>;\
        <a-circumflex>;\
        <ae-ligature>;\
        <c-cedilla>;\
        <e-grave>;\
        <e-acute>;\

```

```

<e-circumflex>;\
<e-diaeresis>;\
<i-circumflex>;\
<i-diaeresis>;\
<o-circumflex>;\
<u-grave>;\
<u-circumflex>;\
<u-diaeresis>

space <tab>;<newline>;<vertical-tab>;<form-feed>;\
<carriage-return>;<space>

cntrl <NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;\
<alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
<form-feed>;<carriage-return>;\
<SO>;<SI>;<DLE>;<DC1>;<DC2>;<DC3>;<DC4>;<NAK>;<SYN>;\
<ETB>;<CAN>;<EM>;<SUB>;<ESC>;<IS4>;<IS3>;<IS2>;\
<IS1>;<DEL>;\
<PAD>;<HOP>;<BPH>;<NBH>;<IND>;<NEL>;<SSA>;<ESA>;\
<HTS>;<HTJ>;<VTS>;<PLD>;<PLU>;<RI>;<SS2>;<SS3>;\
<DCS>;<PU1>;<PU2>;<STS>;<CCH>;<MW>;<SPA>;<EPA>;\
<SOS>;<SGCI>;<SCI>;<CSI>;<ST>;<OSC>;<PM>;<APC>

graph <exclamation-mark>;<quotation-mark>;<number-sign>;\
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;\
<left-parenthesis>;<right-parenthesis>;<asterisk>;<plus-sign>;\
<comma>;<hyphen>;<period>;<slash>;\
<zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;<eight>;<nine>;\
<colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
<greater-than-sign>;<question-mark>;<commercial-at>;\
<A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
<N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>;\
<left-square-bracket>;<backslash>;<right-square-bracket>;\
<circumflex>;<underscore>;<grave-accent>;\
<a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
<n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>;\
<left-brace>;<vertical-line>;<right-brace>;<tilde>;\
<inverted-exclamation-mark>;<cent>;<sterling>;<currency>;<yen>;\
<broken-bar>;<section>;<diaeresis>;<copyright>;<feminine>;\
<guillemot-left>;<not>;<dash>;<registered>;<macron>;\
<degree>;<plus-minus>;<superscript-two>;<superscript-three>;\
<acute>;<mu>;<paragraph>;<dot>;<cedilla>;<superscript-one>;\
<masculine>;<guillemot-right>;<one-quarter>;<one-half>;\
<three-quarters>;<inverted-question-mark>;\
<A-grave>;<A-acute>;<A-circumflex>;<A-tilde>;<A-diaeresis>;\
<A-ring>;<AE-ligature>;<C-cedilla>;<E-grave>;<E-acute>;<E-circumflex>;\
<E-diaeresis>;<I-grave>;<I-acute>;<I-circumflex>;<I-diaeresis>;\
<ETH-icelandic>;<N-tilde>;<O-grave>;<O-acute>;<O-circumflex>;<O-tilde>;\
<O-diaeresis>;<multiplication>;<O-slash>;<U-grave>;<U-acute>;\
<U-circumflex>;<U-diaeresis>;<Y-acute>;<THORN-icelandic>;<s-sharp>;\
<a-grave>;<a-acute>;<a-circumflex>;<a-tilde>;<a-diaeresis>;\
<a-ring>;<ae-ligature>;<c-cedilla>;<e-grave>;<e-acute>;<e-circumflex>;\
<e-diaeresis>;<i-grave>;<i-acute>;<i-circumflex>;<i-diaeresis>;\
<eth-icelandic>;<n-tilde>;<o-grave>;<o-acute>;<o-circumflex>;<o-tilde>;\
<o-diaeresis>;<division>;<o-slash>;<u-grave>;<u-acute>;\
<u-circumflex>;<u-diaeresis>;<y-acute>;<thorn-icelandic>;<y-diaeresis>

print <exclamation-mark>;<quotation-mark>;<number-sign>;\
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;\
<left-parenthesis>;<right-parenthesis>;<asterisk>;<plus-sign>;\
<comma>;<hyphen>;<period>;<slash>;\
<zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;<eight>;<nine>;\
<colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
<greater-than-sign>;<question-mark>;<commercial-at>;\
<A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\

```

```

<N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>;\
<left-square-bracket>;<backslash>;<right-square-bracket>;\
<circumflex>;<underscore>;<grave-accent>;\
<a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
<n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>;\
<left-brace>;<vertical-line>;<right-brace>;<tilde>;\
<inverted-exclamation-mark>;<cent>;<sterling>;<currency>;<yen>;\
<broken-bar>;<section>;<diaeresis>;<copyright>;<feminine>;\
<guillemot-left>;<not>;<dash>;<registered>;<macron>;\
<degree>;<plus-minus>;<superscript-two>;<superscript-three>;\
<acute>;<mu>;<paragraph>;<dot>;<cedilla>;<superscript-one>;\
<masculine>;<guillemot-right>;<one-quarter>;<one-half>;\
<three-quarters>;<inverted-question-mark>;\
<A-grave>;<A-acute>;<A-circumflex>;<A-tilde>;<A-diaeresis>;\
<A-ring>;<AE-ligature>;<C-cedilla>;<E-grave>;<E-acute>;<E-circumflex>;\
<E-diaeresis>;<I-grave>;<I-acute>;<I-circumflex>;<I-diaeresis>;\
<ETH-icelandic>;<N-tilde>;<O-grave>;<O-acute>;<O-circumflex>;<O-tilde>;\
<O-diaeresis>;<multiplication>;<O-slash>;<U-grave>;<U-acute>;\
<U-circumflex>;<U-diaeresis>;<Y-acute>;<THORN-icelandic>;<s-sharp>;\
<a-grave>;<a-acute>;<a-circumflex>;<a-tilde>;<a-diaeresis>;\
<a-ring>;<ae-ligature>;<c-cedilla>;<e-grave>;<e-acute>;<e-circumflex>;\
<e-diaeresis>;<i-grave>;<i-acute>;<i-circumflex>;<i-diaeresis>;\
<eth-icelandic>;<n-tilde>;<o-grave>;<o-acute>;<o-circumflex>;<o-tilde>;\
<o-diaeresis>;<division>;<o-slash>;<u-grave>;<u-acute>;\
<u-circumflex>;<u-diaeresis>;<y-acute>;<thorn-icelandic>;<y-diaeresis>;\
<space>

punct <exclamation-mark>;<quotation-mark>;<number-sign>;\
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;\
<left-parenthesis>;<right-parenthesis>;<asterisk>;\
<plus-sign>;<comma>;<hyphen>;<period>;<slash>;\
<colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
<greater-than-sign>;<question-mark>;<commercial-at>;\
<left-square-bracket>;<backslash>;<right-square-bracket>;\
<circumflex>;<underscore>;<grave-accent>;<left-brace>;\
<vertical-line>;<right-brace>;<tilde>

digit <zero>;<one>;<two>;<three>;<four>;\
<five>;<six>;<seven>;<eight>;<nine>

xdigit <zero>;<one>;<two>;<three>;<four>;\
<five>;<six>;<seven>;<eight>;<nine>;\
<A>;<B>;<C>;<D>;<E>;<F>;\
<a>;<b>;<c>;<d>;<e>;<f>

blank <space>;<tab>

toupper (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
(<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
(<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
(<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
(<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
(<z>,<Z>);\
(<a-grave>,<A-grave>);\
(<a-circumflex>,<A-circumflex>);\
(<ae-ligature>,<AE-ligature>);\
(<c-cedilla>,<C-cedilla>);\
(<e-grave>,<E-grave>);\
(<e-acute>,<E-acute>);\
(<e-circumflex>,<E-circumflex>);\
(<e-diaeresis>,<E-diaeresis>);\
(<i-circumflex>,<I-circumflex>);\
(<i-diaeresis>,<I-diaeresis>);\
(<o-circumflex>,<O-circumflex>);\
(<u-grave>,<U-grave>);\

```

```

(<u-circumflex>,<U-circumflex>);\
(<u-diaeresis>,<U-diaeresis>)

tolower (<A>,<a>);(<B>,<b>);(<C>,<c>);(<D>,<d>);(<E>,<e>);\
(<F>,<f>);(<G>,<g>);(<H>,<h>);(<I>,<i>);(<J>,<j>);\
(<K>,<k>);(<L>,<l>);(<M>,<m>);(<N>,<n>);(<O>,<o>);\
(<P>,<p>);(<Q>,<q>);(<R>,<r>);(<S>,<s>);(<T>,<t>);\
(<U>,<u>);(<V>,<v>);(<W>,<w>);(<X>,<x>);(<Y>,<y>);\
(<Z>,<z>);\
(<A-grave>,<a-grave>);\
(<A-circumflex>,<a-circumflex>);\
(<AE-ligature>,<ae-ligature>);\
(<C-cedilla>,<c-cedilla>);\
(<E-grave>,<e-grave>);\
(<E-acute>,<e-acute>);\
(<E-circumflex>,<e-circumflex>);\
(<E-diaeresis>,<e-diaeresis>);\
(<I-circumflex>,<i-circumflex>);\
(<I-diaeresis>,<i-diaeresis>);\
(<O-circumflex>,<o-circumflex>);\
(<U-grave>,<u-grave>);\
(<U-circumflex>,<u-circumflex>);\
(<U-diaeresis>,<u-diaeresis>)

END LC_CTYPE

#####
LC_COLLATE
#####
#
# The order is control characters, followed by punctuation
# and digits, and then letters. The letters have a
# multi-level sort with diacritics and case being
# ignored on the first pass, then diacritics being
# significant on the second pass, and then case being
# significant on the third (last) pass.
#
order_start                forward;backward;forward

<NUL>
<SOH>
<STX>
<ETX>
<EOT>
<ENQ>
<ACK>
<alert>
<backspace>
<tab>
<newline>
<vertical-tab>
<form-feed>
<carriage-return>
<SO>
<SI>
<DLE>
<DC1>
<DC2>
<DC3>
<DC4>
<NAK>
<SYN>
<ETB>
<CAN>

```



|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <EM>                |                                                             |
| <SUB>               |                                                             |
| <ESC>               |                                                             |
| <IS4>               |                                                             |
| <IS3>               |                                                             |
| <IS2>               |                                                             |
| <IS1>               |                                                             |
| <PAD>               |                                                             |
| <HOP>               |                                                             |
| <BPH>               |                                                             |
| <NBH>               |                                                             |
| <IND>               |                                                             |
| <NEL>               |                                                             |
| <SSA>               |                                                             |
| <ESA>               |                                                             |
| <HTS>               |                                                             |
| <HTJ>               |                                                             |
| <VTS>               |                                                             |
| <PLD>               |                                                             |
| <PLU>               |                                                             |
| <RI>                |                                                             |
| <SS2>               |                                                             |
| <SS3>               |                                                             |
| <DCS>               |                                                             |
| <PU1>               |                                                             |
| <PU2>               |                                                             |
| <STS>               |                                                             |
| <CCH>               |                                                             |
| <MW>                |                                                             |
| <SPA>               |                                                             |
| <EPA>               |                                                             |
| <SOS>               |                                                             |
| <SGCI>              |                                                             |
| <SCI>               |                                                             |
| <CSI>               |                                                             |
| <ST>                |                                                             |
| <OSC>               |                                                             |
| <PM>                |                                                             |
| <APC>               |                                                             |
| <space>             | <space>;<space>;<space>                                     |
| <exclamation-mark>  | <exclamation-mark>;<exclamation-mark>;<exclamation-mark>    |
| <quotation-mark>    | <quotation-mark>;<quotation-mark>;<quotation-mark>          |
| <number-sign>       | <number-sign>;<number-sign>;<number-sign>                   |
| <dollar-sign>       | <dollar-sign>;<dollar-sign>;<dollar-sign>                   |
| <percent-sign>      | <percent-sign>;<percent-sign>;<percent-sign>                |
| <ampersand>         | <ampersand>;<ampersand>;<ampersand>                         |
| <apostrophe>        | <apostrophe>;<apostrophe>;<apostrophe>                      |
| <left-parenthesis>  | <left-parenthesis>;<left-parenthesis>;<left-parenthesis>    |
| <right-parenthesis> | <right-parenthesis>;<right-parenthesis>;<right-parenthesis> |
| <asterisk>          | <asterisk>;<asterisk>;<asterisk>                            |
| <plus-sign>         | <plus-sign>;<plus-sign>;<plus-sign>                         |
| <comma>             | <comma>;<comma>;<comma>                                     |
| <hyphen-minus>      | <hyphen-minus>;<hyphen-minus>;<hyphen-minus>                |
| <period>            | <period>;<period>;<period>                                  |
| <slash>             | <slash>;<slash>;<slash>                                     |
| <zero>              | <zero>;<zero>;<zero>                                        |
| <one>               | <one>;<one>;<one>                                           |
| <two>               | <two>;<two>;<two>                                           |
| <three>             | <three>;<three>;<three>                                     |
| <four>              | <four>;<four>;<four>                                        |
| <five>              | <five>;<five>;<five>                                        |
| <six>               | <six>;<six>;<six>                                           |
| <seven>             | <seven>;<seven>;<seven>                                     |
| <eight>             | <eight>;<eight>;<eight>                                     |
| <nine>              | <nine>;<nine>;<nine>                                        |

|                             |                                                                                     |
|-----------------------------|-------------------------------------------------------------------------------------|
| <colon>                     | <colon>;<colon>;<colon>                                                             |
| <semicolon>                 | <semicolon>;<semicolon>;<semicolon>                                                 |
| <less-than-sign>            | <less-than-sign>;<less-than-sign>;<less-than-sign>                                  |
| <equals-sign>               | <equals-sign>;<equals-sign>;<equals-sign>                                           |
| <greater-than-sign>         | <greater-than-sign>;<greater-than-sign>;<greater-than-sign>                         |
| <question-mark>             | <question-mark>;<question-mark>;<question-mark>                                     |
| <commercial-at>             | <commercial-at>;<commercial-at>;<commercial-at>                                     |
| <left-square-bracket>       | <left-square-bracket>;<left-square-bracket>;<left-square-bracket>                   |
| <backslash>                 | <backslash>;<backslash>;<backslash>                                                 |
| <right-square-bracket>      | <right-square-bracket>;<right-square-bracket>;<right-square-bracket>                |
| <circumflex>                | <circumflex>;<circumflex>;<circumflex>                                              |
| <underscore>                | <underscore>;<underscore>;<underscore>                                              |
| <grave-accent>              | <grave-accent>;<grave-accent>;<grave-accent>                                        |
| <left-brace>                | <left-brace>;<left-brace>;<left-brace>                                              |
| <vertical-line>             | <vertical-line>;<vertical-line>;<vertical-line>                                     |
| <right-brace>               | <right-brace>;<right-brace>;<right-brace>                                           |
| <tilde>                     | <tilde>;<tilde>;<tilde>                                                             |
| <DEL>                       | <DEL>;<DEL>;<DEL>                                                                   |
| <nobreakspace>              | <nobreakspace>;<nobreakspace>;<nobreakspace>                                        |
| <inverted-exclamation-mark> | \                                                                                   |
| <inverted-exclamation-mark> | <inverted-exclamation-mark>;<inverted-exclamation-mark>;<inverted-exclamation-mark> |
| <cent>                      | <cent>;<cent>;<cent>                                                                |
| <sterling>                  | <sterling>;<sterling>;<sterling>                                                    |
| <currency>                  | <currency>;<currency>;<currency>                                                    |
| <yen>                       | <yen>;<yen>;<yen>                                                                   |
| <broken-bar>                | <broken-bar>;<broken-bar>;<broken-bar>                                              |
| <paragraph>                 | <paragraph>;<paragraph>;<paragraph>                                                 |
| <diaeresis>                 | <diaeresis>;<diaeresis>;<diaeresis>                                                 |
| <copyright>                 | <copyright>;<copyright>;<copyright>                                                 |
| <guillemot-left>            | <guillemot-left>;<guillemot-left>;<guillemot-left>                                  |
| <not>                       | <not>;<not>;<not>                                                                   |
| <dash>                      | <dash>;<dash>;<dash>                                                                |
| <registered>                | <registered>;<registered>;<registered>                                              |
| <macron>                    | <macron>;<macron>;<macron>                                                          |
| <degree>                    | <degree>;<degree>;<degree>                                                          |
| <plus-minus>                | <plus-minus>;<plus-minus>;<plus-minus>                                              |
| <superscript-two>           | <two>;<superscript-two>;<superscript-two>                                           |
| <superscript-three>         | <three>;<superscript-three>;<superscript-three>                                     |
| <acute>                     | <acute>;<acute>;<acute>                                                             |
| <mu>                        | <mu>;<mu>;<mu>                                                                      |
| <section>                   | <section>;<section>;<section>                                                       |
| <dot>                       | <dot>;<dot>;<dot>                                                                   |
| <cedilla>                   | <cedilla>;<cedilla>;<cedilla>                                                       |
| <superscript-one>           | <one>;<superscript-one>;<superscript-one>                                           |
| <guillemot-right>           | <guillemot-right>;<guillemot-right>;<guillemot-right>                               |
| <one-quarter>               | <zero>;<one-quarter>;<one-quarter>                                                  |
| <one-half>                  | <zero>;<one-half>;<one-half>                                                        |
| <three-quarters>            | <zero>;<three-quarters>;<three-quarters>                                            |
| <inverted-question-mark>    | \                                                                                   |
| <inverted-question-mark>    | <inverted-question-mark>;<inverted-question-mark>;<inverted-question-mark>          |
| <multiplication>            | <multiplication>;<multiplication>;<multiplication>                                  |
| <division>                  | <division>;<division>;<division>                                                    |
| <a>                         | <a>;<a>;<a>                                                                         |
| <A>                         | <a>;<a>;<A>                                                                         |
| <feminine>                  | <a>;<feminine>;<feminine>                                                           |
| <a-acute>                   | <a>;<a-acute>;<a-acute>                                                             |
| <A-acute>                   | <a>;<a-acute>;<A-acute>                                                             |
| <a-grave>                   | <a>;<a-grave>;<a-grave>                                                             |
| <A-grave>                   | <a>;<a-grave>;<A-grave>                                                             |
| <a-circumflex>              | <a>;<a-circumflex>;<a-circumflex>                                                   |
| <A-circumflex>              | <a>;<a-circumflex>;<A-circumflex>                                                   |
| <a-ring>                    | <a>;<a-ring>;<a-ring>                                                               |
| <A-ring>                    | <a>;<a-ring>;<A-ring>                                                               |
| <a-diaeresis>               | <a>;<a-diaeresis>;<a-diaeresis>                                                     |
| <A-diaeresis>               | <a>;<a-diaeresis>;<A-diaeresis>                                                     |

|                 |                                     |
|-----------------|-------------------------------------|
| <a-tilde>       | <a>;<a-tilde>;<a-tilde>             |
| <A-tilde>       | <a>;<a-tilde>;<A-tilde>             |
| <ae-ligature>   | <a>;<a><e>;<a><e>                   |
| <AE-ligature>   | <a>;<a><e>;<A><E>                   |
| <b>             | <b>;<b>;<b>                         |
| <B>             | <b>;<b>;<B>                         |
| <c>             | <c>;<c>;<c>                         |
| <C>             | <c>;<c>;<C>                         |
| <c-cedilla>     | <c>;<c-cedilla>;<c-cedilla>         |
| <C-cedilla>     | <c>;<c-cedilla>;<C-cedilla>         |
| <d>             | <d>;<d>;<d>                         |
| <D>             | <d>;<d>;<D>                         |
| <eth-icelandic> | <d>;<eth-icelandic>;<eth-icelandic> |
| <ETH-icelandic> | <d>;<eth-icelandic>;<ETH-icelandic> |
| <e>             | <e>;<e>;<e>                         |
| <E>             | <e>;<e>;<E>                         |
| <e-acute>       | <e>;<e-acute>;<e-acute>             |
| <E-acute>       | <e>;<e-acute>;<E-acute>             |
| <e-grave>       | <e>;<e-grave>;<e-grave>             |
| <E-grave>       | <e>;<e-grave>;<E-grave>             |
| <e-circumflex>  | <e>;<e-circumflex>;<e-circumflex>   |
| <E-circumflex>  | <e>;<e-circumflex>;<E-circumflex>   |
| <e-diaeresis>   | <e>;<e-diaeresis>;<e-diaeresis>     |
| <E-diaeresis>   | <e>;<e-diaeresis>;<E-diaeresis>     |
| <f>             | <f>;<f>;<f>                         |
| <F>             | <f>;<f>;<F>                         |
| <g>             | <g>;<g>;<g>                         |
| <G>             | <g>;<g>;<G>                         |
| <h>             | <h>;<h>;<h>                         |
| <H>             | <h>;<h>;<H>                         |
| <i>             | <i>;<i>;<i>                         |
| <I>             | <i>;<i>;<I>                         |
| <i-acute>       | <i>;<i-acute>;<i-acute>             |
| <I-acute>       | <i>;<i-acute>;<I-acute>             |
| <i-grave>       | <i>;<i-grave>;<i-grave>             |
| <I-grave>       | <i>;<i-grave>;<I-grave>             |
| <i-circumflex>  | <i>;<i-circumflex>;<i-circumflex>   |
| <I-circumflex>  | <i>;<i-circumflex>;<I-circumflex>   |
| <i-diaeresis>   | <i>;<i-diaeresis>;<i-diaeresis>     |
| <I-diaeresis>   | <i>;<i-diaeresis>;<I-diaeresis>     |
| <j>             | <j>;<j>;<j>                         |
| <J>             | <j>;<j>;<J>                         |
| <k>             | <k>;<k>;<k>                         |
| <K>             | <k>;<k>;<K>                         |
| <l>             | <l>;<l>;<l>                         |
| <L>             | <l>;<l>;<L>                         |
| <m>             | <m>;<m>;<m>                         |
| <M>             | <m>;<m>;<M>                         |
| <n>             | <n>;<n>;<n>                         |
| <N>             | <n>;<n>;<N>                         |
| <n-tilde>       | <n>;<n-tilde>;<n-tilde>             |
| <N-tilde>       | <n>;<n-tilde>;<N-tilde>             |
| <o>             | <o>;<o>;<o>                         |
| <O>             | <o>;<o>;<O>                         |
| <masculine>     | <o>;<masculine>;<masculine>         |
| <o-acute>       | <o>;<o-acute>;<o-acute>             |
| <O-acute>       | <o>;<o-acute>;<O-acute>             |
| <o-grave>       | <o>;<o-grave>;<o-grave>             |
| <O-grave>       | <o>;<o-grave>;<O-grave>             |
| <o-circumflex>  | <o>;<o-circumflex>;<o-circumflex>   |
| <O-circumflex>  | <o>;<o-circumflex>;<O-circumflex>   |
| <o-diaeresis>   | <o>;<o-diaeresis>;<o-diaeresis>     |
| <O-diaeresis>   | <o>;<o-diaeresis>;<O-diaeresis>     |
| <o-tilde>       | <o>;<o-tilde>;<o-tilde>             |
| <O-tilde>       | <o>;<o-tilde>;<O-tilde>             |

```

<o-slash>          <o>;<o-slash>;<o-slash>
<O-slash>          <o>;<o-slash>;<O-slash>
<p>                 <p>;<p>;<p>
<P>                 <p>;<p>;<P>
<q>                 <q>;<q>;<q>
<Q>                 <q>;<q>;<Q>
<r>                 <r>;<r>;<r>
<R>                 <r>;<r>;<R>
<s>                 <s>;<s>;<s>
<S>                 <s>;<s>;<S>
<s-sharp>           <s>;<s><s>;<s><s>
<t>                 <t>;<t>;<t>
<T>                 <t>;<t>;<T>
<thorn-icelandic> <t>;<t><h>;<t><h>
<THORN-icelandic> <t>;<t><h>;<T><h>
<u>                 <u>;<u>;<u>
<U>                 <u>;<u>;<U>
<u-acute>           <u>;<u-acute>;<u-acute>
<U-acute>           <u>;<u-acute>;<U-acute>
<u-grave>           <u>;<u-grave>;<u-grave>
<U-grave>           <u>;<u-grave>;<U-grave>
<u-circumflex>      <u>;<u-circumflex>;<u-circumflex>
<U-circumflex>      <u>;<u-circumflex>;<U-circumflex>
<u-diaeresis>       <u>;<u-diaeresis>;<u-diaeresis>
<U-diaeresis>       <u>;<u-diaeresis>;<U-diaeresis>
<v>                 <v>;<v>;<v>
<V>                 <v>;<v>;<V>
<w>                 <w>;<w>;<w>
<W>                 <w>;<w>;<W>
<x>                 <x>;<x>;<x>
<X>                 <x>;<x>;<X>
<y>                 <y>;<y>;<y>
<Y>                 <y>;<y>;<Y>
<y-acute>           <y>;<y-acute>;<y-acute>
<Y-acute>           <y>;<y-acute>;<Y-acute>
<y-diaeresis>       <y>;<y-diaeresis>;<y-diaeresis>
<Z>                 <z>;<z>;<z>
<Z>                 <z>;<z>;<Z>
UNDEFINED
order_end

END LC_COLLATE

#####
LC_MONETARY
#####

int_curr_symbol      "<F><R><F><space>"
currency_symbol      "<F>"
mon_decimal_point    "<comma>"
mon_thousands_sep    ""
mon_grouping         3;0
positive_sign        ""
negative_sign         "<hyphen>"
int_frac_digits      2
frac_digits          2
p_cs_precedes         0
p_sep_by_space        1
n_cs_precedes         0
n_sep_by_space        1
p_sign_posn           1
n_sign_posn           1

END LC_MONETARY

```

```

#####
LC_NUMERIC
#####

decimal_point      "<comma>"
thousands_sep      ""
grouping            3;0

END LC_NUMERIC

#####
LC_TIME
#####
# abbreviated day names
abday      "<d><i><m>"\
           "<l><u><n>"\
           "<m><a><r>"\
           "<m><e><r>"\
           "<j><e><u>"\
           "<v><e><n>"\
           "<s><a><m>"

# full day names
day      "<d><i><m><a><n><c><h><e>"\
         "<l><u><n><d><i>"\
         "<m><a><r><d><i>"\
         "<m><e><r><c><r><e><d><i>"\
         "<j><e><u><d><i>"\
         "<v><e><n><d><r><e><d><i>"\
         "<s><a><m><e><d><i>"

# abbreviated month names
abmon    "<j><a><n>"\
         "<f><e><acute><v>"\
         "<m><a><r>"\
         "<a><v><r>"\
         "<m><a><i>"\
         "<j><u><n>"\
         "<j><u><l>"\
         "<a><o><u><circumflex>"\
         "<s><e><p>"\
         "<o><c><t>"\
         "<n><o><v>"\
         "<d><e><acute><c>"

# full month names
mon      "<j><a><n><v><i><e><r>"\
         "<f><e><acute><v><r><i><e><r>"\
         "<m><a><r><s>"\
         "<a><v><r><i><l>"\
         "<m><a><i>"\
         "<j><u><i><n>"\
         "<j><u><i><l><l><e><t>"\
         "<a><o><u><circumflex><t>"\
         "<s><e><p><t><e><m><b><r><e>"\
         "<o><c><t><o><b><r><e>"\
         "<n><o><v><e><m><b><r><e>"\
         "<d><e><acute><c><e><m><b><r><e>"

# date/time format. The following designates this
# format: "%a %e %b %H:%M:%S %Z %Y"
d_t_fmt  "<percent-sign><a><space><percent-sign><e>\

```

```

<space><percent-sign><b><space><percent-sign><H>\
<colon><percent-sign><M><colon><percent-sign><S>\
<space><percent-sign><Z><space><percent-sign><Y>"

# date format. The following designates this
# format: "%d.%m.%y"
d_fmt    "<percent-sign><d><period><percent-sign><m>\
<period><percent-sign><y>"

# time format. The following designates this
# format: "%H:%M:%S"
t_fmt    "<percent-sign><H><colon><percent-sign><M>\
<colon><percent-sign><S>"

am_pm    "<semicolon>"

# 12-hour time representation. This is empty, meaning
# this locale always uses 24-hour format.
t_fmt_ampm    ""

END LC_TIME

```

```

#####
LC_MESSAGES
#####

# yes expression. The following designates:
# "^( [oO] | [oO] [uU] [iI] ) "
yesexpr  "<circumflex><left-parenthesis>\
<left-square-bracket><o><O><right-square-bracket>\
<vertical-line><left-square-bracket><o><O>\
<right-square-bracket><left-square-bracket><u><U>\
<right-square-bracket><left-square-bracket><i><I>\
<right-square-bracket><right-parenthesis>"

# no expression. The following designates:
# "^( [nN] | [nN] [oO] [nN] ) "
noexpr   "<circumflex><left-parenthesis>\
<left-square-bracket><n><N><right-square-bracket>\
<vertical-line><left-square-bracket><n><N>\
<right-square-bracket><left-square-bracket><o><O>\
<right-square-bracket><left-square-bracket><n><N>\
<right-square-bracket><right-parenthesis>"

# yes string. The following designates: "oui:o:O"
yesstr   "<o><u><i><colon><o><colon><O>"

# no string. The following designates: "non:n:N"
nostr    "<n><o><n><colon><n><colon><N>"

END LC_MESSAGES

```

---

## 用語集

### ASCII

情報交換用米国標準コード。ASCII は、制御文字とグラフィック文字を含み、7 ビットのバイナリ値で表現される 128 文字を定義している (ISO 646 も参照)。

文字セット (*character set*)、コード化文字セット (*coded character set*) も参照

### C ロケール (C locale)

標準の (省略時の) 言語環境。この環境は、非国際化アプリケーション用として、ロケールがインストールされていない場合やロケールがアクティブでない場合でも、必ず利用できる。

### dense コード (dense code)

本オペレーティング・システムは、dense コードと Unicode の 2 つのタイプのロケールをサポートする。dense コード・ロケールは、コードポイントを連続で割り当てて空き位置をなくすことによってテーブル・サイズを最小にする、ワイド文字エンコーディングを使用する。dense コード・ロケールでは、あるロケールの `wchar_t` 値は、他のロケールで同じ文字を指すとは限らない。つまり、`wchar_t` 値はロケールごとに定義される。

Unicode も参照

### I18N

国際化 (*internationalization*) を参照

### ISO 10646

ISO ユニバーサル文字セット (UCS)。この文字セットの最初の 65,536 コード位置は基本多言語プレーン (BMP) と呼ばれ、各文字は 16 ビット長である。ISO 10646 のこの形式は、UCS-2 と呼ばれる。ISO 10646 には、各文字が 32 ビット長である、UCS-4 と呼ばれる形式もある。

Unicode も参照

### ISO 646

情報交換用の ISO 7 ビット・コードセット。ISO 646 の参照バージョンは、ASCII コードセットで定義されているグラフィック文字と同一の、95 個のグラフィック文字を含む。

### **ISO 6937**

公衆通信網，私設通信網，または磁気テープや磁気ディスクなどの交換メディアを使用したテキスト通信用の ISO 7 ビットまたは 8 ビットのコードセット。

### **ISO8859-\***

ISO 8 ビット・シングルバイト・コードセット。アスタリスク (\*) は，関連する ISO 標準のパートを表す番号を示す。たとえば，ISO8859-1 コードセットは，大半の西ヨーロッパ言語の要件を満たす 191 個のグラフィック文字を定義している ISO 8859 パート 1，ラテン・アルファベット 1 に適合する。

### **L10N**

地域化 (*localization*) を参照

### **langinfo データベース (langinfo database)**

ロケールの数値，通貨，日付，時刻，およびメッセージに関連する情報の集まり。

### **UCS**

ISO 10646 を参照

### **Unicode**

大半の母国語の文字に対するエンコーディングを定義する規格。Unicode 標準は，UCS (Universal Character Set) を規定し，多数の文字を定義します。この中には，ベンダ定義文字用のプライベート用領域も含まれる。“Unicode”の元の意味は，ISO 10646 規格で定義されている UCS-2 (16 ビット) エンコーディングに限定されたエンコーディング。現在 Unicode 標準は，UCS-4 (32 ビット) エンコーディングを含み，データ・ファイルを処理するバイト指向のプロトコルで使用する，多数の UTF (Universal Transformation Format) を定義するようになった。

コード化文字セット (*coded character set*)，ISO 10646 も参照

### **グラフィック文字 (graphic character)**

手書き，印字，あるいは表示する際に，視覚的に表現される制御文字以外の文字。表意文字ともいいます。

### **コード化文字セット (coded character set)**

文字セットを定義し，そのセット中の各文字とそのビット表現との 1 対 1 の関係を明確に定義する規則の集合。UNIX システム上では，コードセットという用語が一般的。MS-DOS および Microsoft Windows システムでは，コード・ページという用語が一般的。



**コードセット (code set)**

コード化文字セット (*coded character set*) を参照

**コード・ページ (code page)**

コード化文字セット (*coded character set*) を参照

**国際化 (internationalization)**

プログラムが扱う言語や、文化的データ、文字のエンコード方式についてあらかじめ知らなくても、プログラムの開発が行えるようにする仕組み。国際化プログラムは、実行時に特定の言語環境に合わせて動作を変更するためのインタフェース群を使用する。I18N という用語が、国際化 (internationalization) の略語としてよく使用される。

ロケール (*locale*)、地域化 (*localization*) も参照

**照合順序 (collating sequence)**

ソート時に文字または文字グループに適用される順序付け規則。

**小数点文字 (radix character)**

数の整数部と小数部を区切る文字。

**処理コード (process code)**

プログラム内のデータを操作するために使用されるエンコード方式。

ファイル・コードと対比

**制御文字 (control character)**

テキストの記録や、処理、転送、解釈に影響する、グラフィック文字以外の文字。

**地域 (territory)**

一般的に民族や国などの政治的な実体によって定義される、地理的な領域。地域化で対処しなければならない文化的な相違 (たとえば、地域の通貨や言語) がある。

**地域化 (localization)**

コンピュータ・システムに、言語や文化固有の情報を組み込むための仕組み。そのための要件のいくつかは、ロケールによって解決される。他の要件は、プログラム・メッセージを翻訳し、プリンタ・デバイスとディスプレイ・デバイスに適切なフォントを搭載することにより、また場合によっては、さらに追加ソフトウェアを作成することにより実現される。L10N という用語が、地域化 (localization) の略語として使用されることがある。

国際化 (*internationalization*) , ロケール (*locale*) も参照

#### データ (**data**)

内部的に生成された情報 , ファイルから取り出された情報 , ファイルへ書き込んだ情報 , プログラムのユーザと通信するためのメッセージ・テキストを指す。

#### ファイル・コード (**file code**)

プログラム外のデータに適用されるエンコード方式。

#### 処理コードと対比

#### 符号拡張 (**sign extension**)

小さなデータ型の値は , 大きなデータ型と比較するために変換される場合 , 残りの上位ビットに最上位ビットが埋め込まれる。たとえば , `s[0]` は , 値が `0x8e` の場合 , 符号拡張により `0xffff8e` として扱われる。

#### 文化的データ (**cultural data**)

日付 , 時刻 , 数値 , 金額値などの地域固有の表記。

#### 母国語 (**native language**)

英語 , フランス語 , 日本語 , タイ語などの , コンピュータ・ユーザが話したり書いたりする言語。

#### マルチバイト文字 (**multibyte character**)

文字 (*character*) を参照

#### メッセージ・カタログ (**message catalog**)

特定の言語 , 地域 , およびコードセットの組み合わせで使用される , プログラム・メッセージ , コマンド・プロンプト , およびプロンプトへの応答を含む , プログラム・コード外部のファイルまたは記憶領域。

#### 文字 (**character**)

1 つのグラフィック・シンボルまたは制御コードを表現する , 1 つ以上のバイトの並び。C の `char` データ型とは異なり , 文字は 1 バイトまたは複数バイトの値で表現される。「マルチバイト文字」という表現は , 「文字」という用語と同義である。つまり両者とも , 1 バイト値を含む , 任意の長さの文字値を意味する。

ワイド文字 (*wide character*) も参照

### 文字セット (character set)

テキストの構成，制御，または表現に使用される要素セットのメンバ。

*ASCII*，*ISO 10646* も参照

### 文字の列 (character string)

ヌル・バイトで終わる，連続したバイト (ヌル・バイトも含む) の並び。C プログラミング言語では，文字列は `char` 型の配列である。ヌル・バイトは，すべてのビットがゼロ (0) のバイトである。

空の文字列は，1 番目の要素がヌル・バイトの文字列である。

文字 (*character*)，ワイド文字列 (*wide-character string*) も参照

### 文字列 (string)

文字の列 (*character string*) を参照

### ユーロ (euro)

EMU (Economic and Monetary Union) に所属するヨーロッパ諸国に導入された新しい通貨。西暦 2002 年に，EMU に属する国々の通貨は，この新しい通貨と置き換えられます。ユーロは，等号 (=) と大文字の C で構成される通貨記号を持ち，国際的な通貨ドキュメントでは文字列 `EUR` で識別されます。

### ユニバーサル文字セット (Universal Character Set)

*ISO 10646* を参照

### ローカル言語 (local language)

母国語 (*native language*) を参照

### ロケール (locale)

母国語 (ローカル言語)，文化的データ，およびコードセットの特定の組み合わせをサポートするデータおよび規則の集合。言語テーブルともいいます。コード化文字セット (*coded character set*)，文化的データ (*cultural data*)，*langinfo* データベース (*langinfo database*)，地域化 (*localization*) も参照

### ワイド文字 (wide character)

拡張実行文字セットのすべてのメンバを格納するのに十分な大きさの整数型。プログラムの観点からは，ワイド文字は，ヘッダ・ファイル `/usr/include/stddef.h` (X/Open 仕様に準拠) と `/usr/include/stdlib.h` (ANSI C 標準に準拠) で定義されている `wchar_t` 型のオブジェクト。`wchar_t` データ型が定義されているファイル位置は標準化組織が決定するが，定義そのものは実装に固有である。たとえば，

1 バイト・コードセットだけをサポートする実装では、`wchar_t` を 1 バイト値として定義できる。Tru64 UNIX システムでは、`wchar_t` は 4 バイト (32 ビット) 値である。

ヌル・ワイド文字は、すべてのビットがゼロ (0) の `wchar_t` 値。

#### ワイド文字列 (**wide-character string**)

ヌル・ワイド文字で終わる、連続したワイド文字 (ヌル・ワイド文字も含む) の並び。ワイド文字列は、`wchar_t` 型の配列である。

文字の列 (*character string*)、ワイド文字 (*wide character*) も参照

#### ワールドワイド・ポータビリティ・インタフェース (**WPI**)

シングルバイト・コードセットやマルチバイト・コードセットをサポートするアプリケーションを作成するときに使用できる関数。WPI 関数は、C 言語インタフェースと似ているが、ワイド文字を使用する。

## 数字および記号

## 3 文字表記

C 言語コンパイラによるサポート ..... 2-15

## 8 進値

メッセージ文字列 ..... 3-6

## A

**add\_wchnstr** 関数 ..... 4-4  
**add\_wchstr** 関数 ..... 4-4  
**add\_wch** 関数 ..... 4-2  
**addnwstr** マクロ ..... 4-5  
**addwchnstr** マクロ ..... 4-4  
**addwchstr** マクロ ..... 4-4  
**addwch** マクロ ..... 4-2  
**addwstr** マクロ ..... 4-5  
**ASCII** コードセット ..... 2-4  
**asctime** 関数 ..... A-5  
**asort** コマンド ..... 7-6  
   UDC の照合 ..... B-20  
   アプリケーション ..... 7-7  
   オプション ..... 7-7

## B

**BUFSIZE** 定数

定義 ..... 2-31

## C

**CAT\_NAME** 定数

定義 ..... 2-31

**catclose** 関数

**nl\_catd** 記述子型 ..... 2-24  
**NLSPATH** 環境変数 ..... 2-25  
 引数 ..... 3-40  
 メッセージ・カタログへのプログラム  
   ム呼び出し ..... 3-34

**catgets** 関数

**printf** 関数 ..... 2-26  
**puts** 関数 ..... 2-24  
 カタログ・オープン失敗の検  
   出 ..... 3-39  
 動的なコードセット変換 ..... 3-21  
 引数 ..... 3-40  
 プログラム定義のマクロ ..... 3-41  
 メッセージ・カタログへのプログラ  
   ム呼び出し ..... 3-34

**catopen** 関数 ..... 3-34

**NL\_CAT\_LOCALE** 定数 ..... 3-35  
**nl\_catd** 記述子型 ..... 2-24  
**NLSPATH** 環境変数... 2-25, 3-35  
**root** のアカウント下 ..... 3-39  
 エラー・ステータスを返せな  
   い ..... 3-39  
 カタログのパス名検索の例... 3-38  
 コードセット変換のサポート 3-39

|                                 |      |                            |            |
|---------------------------------|------|----------------------------|------------|
| 性能のオーバヘッド.....                  | 3-29 | カーソル・モード.....              | B-12       |
| 動的なコードセット変換.....                | 3-21 | キー・バインドの編集.....            | B-12       |
| トラブルシューティング.....                | 3-38 | 使用バッファ.....                | B-14       |
| 引数.....                         | 3-35 | タイプ・モード.....               | B-13       |
| ヘッダ・ファイルの要件.....                | 3-35 | 取り消しバッファ.....              | B-14       |
| メッセージ・カタログ検索の制<br>御.....        | 3-35 | ビットマップ・データ・バッ<br>ファ.....   | B-13       |
| メッセージ・カタログへのプログラ<br>ム呼び出し.....  | 3-34 | 描画キー.....                  | B-17       |
| <b>cc</b> コマンド                  |      | フォント編集画面.....              | B-9, B-14  |
| 3 文字表記.....                     | 2-15 | フォント編集のファンクション・<br>キー..... | B-15       |
| ロケール・メソッド定義のコンパイ<br>ル.....      | 6-62 | ヘルプ・オプション.....             | B-9        |
| <b>CDE</b>                      |      | 編集機能キー.....                | B-18       |
| UID ファイル.....                   | 5-2  | 編集機能用のキーマップ....            | B-14       |
| スタイルマネージャのバックドロッ<br>プ・ファイル..... | 5-2  | 編集バッファ.....                | B-13       |
| スタイルマネージャのパレット・<br>ファイル.....    | 5-2  | ペースト・モード.....              | B-13       |
| データタイプ・ファイル.....                | 5-2  | モード切り替えキー.....             | B-16       |
| ヘルプ・ファイル.....                   | 5-2  | ラップ・モード.....               | B-13       |
| メッセージ・カタログ.....                 | 5-2  | <b>cedit</b> エディタ          |            |
| リソース・ファイル.....                  | 5-2  | UDC の作成.....               | B-1        |
| リファレンス・ページ.....                 | 5-2  | <b>cedit</b> コマンド.....     | B-3        |
| ロケールのサポートを有効にす<br>る.....        | 5-1  | C オプション.....               | B-4        |
| <b>cedit</b>                    |      | h オプション.....               | B-4        |
| dtterm の問題.....                 | B-5  | r オプション.....               | B-4        |
| Refer 編集機能.....                 | B-14 | オプションと引数.....              | B-4        |
| Scale 編集機能.....                 | B-14 | フォント編集画面....               | B-10, B-14 |
| Use 編集機能.....                   | B-14 | 編集モード.....                 | B-12       |
| カット-アンド-ペースト・バッ<br>ファ.....      | B-14 | メッセージ言語の変更.....            | B-8        |
| カーソル移動キー.....                   | B-17 | メニュー・インタフェース....           | B-5        |
| カーソル制御キー.....                   | B-16 | メニュー項目の状態.....             | B-6        |
|                                 |      | ユーザ・インタフェース画面 .            | B-4        |
|                                 |      | <b>cedit</b> メニュー          |            |
|                                 |      | options メニュー.....          | B-8        |
|                                 |      | UDC オプション.....             | B-7        |
|                                 |      | コマンド・オプション.....            | B-8        |

|                                  |            |                           |      |
|----------------------------------|------------|---------------------------|------|
| 削除オプション .....                    | B-7        | <b>cp_dirs</b> ファイル ..... | 7-4  |
| 表示オプション .....                    | B-7        | bdf エントリ .....            | 7-5  |
| ファイル・オプション .....                 | B-7        | cdb エントリ .....            | 7-5  |
| 編集オプション .....                    | B-7        | iks エントリ .....            | 7-5  |
| <b>cgen</b> コマンド .....           | B-20       | odl エントリ .....            | 7-5  |
| bdf オプション .....                  | B-21, B-25 | pcf エントリ .....            | 7-5  |
| col オプション .....                  | B-21       | pre エントリ .....            | 7-5  |
| fprop オプション .....                | B-21       | sim エントリ .....            | 7-5  |
| iks オプション .....                  | B-21       | udc エントリ .....            | 7-5  |
| merge オプション .....                | B-21       | 省略時のエントリ .....            | 7-4e |
| odl オプション .....                  | B-21       | <b>ctime</b> 関数 .....     | A-5  |
| osiz オプション .....                 | B-21       | <b>curses</b> インタフェース     |      |
| pcf オプション .....                  | B-21, B-25 | 複数からの選択 .....             | 4-1  |
| pre オプション .....                  | B-21       | <b>curses</b> ライブラリ       |      |
| UDC サポート・ファイルの作                  |            | 推奨ルーチン .....              | 4-1  |
| 成 .....                          | B-1        | マルチカラム文字の上書き .....        | 4-1  |
| win オプション .....                  | B-21       | マルチバイト文字 .....            | 4-1  |
| オプション .....                      | B-21       | リファレンス・ページ .....          | 4-1  |
| オプションなし .....                    | B-21       | ワイド文字データ .....            | 4-1  |
| フォント・レンダラとの比較                    | 7-15       | <b>curses</b> ルーチン        |      |
| <b>charmap</b> ファイル .....        | 6-2        | curses ウィンドウからのワイド文字      |      |
| キーワード宣言 .....                    | 6-4        | の削除 .....                 | 4-8  |
| シンボル名の標準化 .....                  | 6-7        | curses ウィンドウからのワイド文字      |      |
| 文字エンコーディング .....                 | 6-4        | の読み取り .....               | 4-9  |
| 文字シンボル .....                     | 6-4, 6-7   | curses ウィンドウでの書式付きテキ      |      |
| <b>char</b> データ型                 |            | ストの表示 .....               | 4-16 |
| ISO C 関数の一覧 .....                | A-1        | curses ウィンドウ内の書式付きテキ      |      |
| <b>Clear Display</b> オプション       |            | ストの変換 .....               | 4-15 |
| DECterm の罫線 .....                | C-5        | カーソルを進めずにワイド文字列を          |      |
| <b>COLL_WEIGHTS_MAX</b> 変数 ..... | 6-16       | 挿入する .....                | 4-7  |
| <b>comment_char</b> キーワード .....  | 6-8        | カーソルを進めてワイド文字を追加          |      |
| <b>copy</b> 文 .....              | 6-9        | する .....                  | 4-2  |
| LC_CTYPE カテゴリの .....             | 6-14       |                           |      |

|                               |      |
|-------------------------------|------|
| カーソルを進めないでワイド文字を<br>挿入 .....  | 4-3  |
| 属性なしでのワイド文字列の読み取<br>り .....   | 4-11 |
| 端末からの文字列の読み取り                 | 4-12 |
| ワイド文字のキーボードからの読み<br>取り .....  | 4-14 |
| ワイド文字列とその属性の読み取<br>り .....    | 4-10 |
| ワイド文字列を追加しカーソルを進<br>めない ..... | 4-4  |
| ワイド文字列を追加しカーソルを進<br>める .....  | 4-5  |
| <b>C コンパイラ</b>                |      |
| 3 文字表記 .....                  | 2-15 |
| <b>C ライブラリ</b>                |      |
| 国際化 .....                     | 2-2  |

## D

|                                       |      |
|---------------------------------------|------|
| <b>D_FMT</b> 定数                       |      |
| strftime 関数 .....                     | 2-21 |
| <b>D_T_FMT</b> 定数                     |      |
| nl_langinfo 関数での使用 ....               | 2-20 |
| <b>DCH</b> エスケープ・シーケンス ..             | C-5  |
| <b>DECCOLM</b> エスケープ・シーケン<br>ス .....  | C-5  |
| <b>DECDHLB</b> エスケープ・シーケン<br>ス .....  | C-5  |
| <b>DECDHLT</b> エスケープ・シーケン<br>ス .....  | C-5  |
| <b>DECDLD</b> 制御文字列 .....             | C-7  |
| <b>DECDRLBR</b> エスケープ・シーケン<br>ス ..... | C-1  |
| パラメータ .....                           | C-3  |

|                                        |      |
|----------------------------------------|------|
| ビット・パターン・マッピング                         | C-2  |
| <b>DECDWL</b> エスケープ・シーケン<br>ス .....    | C-5  |
| <b>DECERLBRA</b> エスケープ・シーケ<br>ンス ..... | C-4  |
| 罫線の削除 .....                            | C-4  |
| <b>DECERLBRP</b> エスケープ・シーケン<br>ス ..... | C-4  |
| <b>DECSTR</b> エスケープ・シーケン<br>ス .....    | C-5  |
| <b>DECterm</b>                         |      |
| Clear Display オプション .....              | C-5  |
| Reset Terminal オプション .....             | C-5  |
| エスケープ・シーケンスと罫線                         | C-5  |
| 罫線サポートの判定 .....                        | C-7  |
| 罫線の開始点 .....                           | C-3  |
| 罫線の削除 .....                            | C-4  |
| 罫線の長さ .....                            | C-4  |
| 罫線のビットマスク・パターン                         | C-3  |
| 罫線の描画 .....                            | C-1  |
| 端末プログラミングの制約 .....                     | C-7  |
| デバイス属性のレポート .....                      | C-7  |
| 領域内の罫線の削除 .....                        | C-4  |
| <b>DECterm</b> の制約                     |      |
| DECDLD 制御文字列 .....                     | C-7  |
| ダウンライン・ローダブル文字                         | C-7  |
| 標準文字セット (SCS) .....                    | C-7  |
| <b>delch</b> マクロ .....                 | 4-8  |
| <b>delset</b> ディレクティブ                  |      |
| 制限事項 .....                             | 3-9  |
| メッセージ・セットの削除 .....                     | 3-9  |
| メッセージ・ソース・ファイルでの<br>位置 .....           | 3-9  |
| <b>dense</b> コード・ロケール .....            | 2-2  |
| Unicode と同じ charmap .....              | 6-19 |



Unicode との類似点 ..... 2-4  
 アプリケーションでの使用 .... 2-3  
 ワイド文字のエンコーディング 2-3  
**DL** エスケープ・シーケンス .... C-5  
**DRCS (DIGITAL Replacement Character Set)** ..... C-7  
**dspcat** コマンド ..... 3-32  
 カタログ出力ストリームの再フォーマット ..... 3-32  
**dspmsg** コマンド ..... 3-32  
 テキスト文字列の置き換え ... 3-33

## E

**echo\_wchar** 関数 ..... 4-2  
**echowchar** マクロ ..... 4-2  
**ECH** エスケープ・シーケンス... C-5  
**ED** エスケープ・シーケンス .... C-5  
**EL** エスケープ・シーケンス ..... C-5  
**errno**  
 スレッドセーフな方法での設定 ..... 6-43  
**escape\_char** キーワード ..... 6-8  
**exit** 関数  
 メッセージ・カタログのクローズ ..... 3-40  
**extract** コマンド ..... 3-18

## F

**fgetc** 関数 ..... A-9  
**fgets** 関数 ..... A-9  
**fgetwc** 関数 ..... A-9  
**fgetws** 関数 ..... A-9

メソッドの作成 ..... 6-30  
**fontconverter** コマンド ..... B-23  
**font** オプション ..... B-25  
**h** オプション ..... B-25  
**merge** オプション ..... B-25  
**merge** オプションとフォント・ファイル・フォーマット ..... B-27  
**preload** オプション ..... B-25  
**udc** オプション ..... B-25  
**w** オプション ..... B-25  
 オプションと引数 ..... B-25  
 代替のコマンド **cgen** ..... B-24  
 フォント・ファイルの省略時の作成 ..... B-27  
**fprintf** 関数 ..... A-6  
**fputs** 関数 ..... A-9  
**fputws** 関数 ..... A-9  
 メソッドの作成 ..... 6-38  
**fscanf** 関数 ..... A-6  
**fwide** 関数 ..... A-9  
**fwprintf** 関数 ..... A-6  
**fwscanf** 関数 ..... A-6

## G

**gencat** コマンド ..... 3-26  
**delset** ディレクティブ ..... 3-9  
**dspcat** コマンド出力の使用 .. 3-32  
**makefile** での使用 ..... 3-23  
**X/Open** による定義 ..... 2-24  
 一般的なエラー ..... 3-26  
 識別子の不注意な変更を避ける ..... 3-27

|                             |      |                               |            |
|-----------------------------|------|-------------------------------|------------|
| 対話型の使用 .....                | 3-22 | <b>iconv_open</b> 関数 .....    | A-12       |
| 複数のソース・ファイルの処               |      | <b>iconv</b> 関数 .....         | 7-11, A-12 |
| 理 .....                     | 3-30 | アルゴリズムに基づくコンバータの              |            |
| 無視される行 .....                | 3-13 | 位置 .....                      | 7-12       |
| メッセージ・カタログの作成               | 3-26 | テーブル・コンバータの位置                 | 7-12       |
| メッセージ・カタログの生成               | 3-22 | 別名ファイル .....                  | 7-12       |
| メッセージ・カタログの変更               | 3-26 | <b>iconv</b> コマンド .....       | 7-11, A-12 |
| メッセージ・セットの削除...             | 3-10 | アルゴリズムに基づくコンバータの              |            |
| メッセージ・ソースの変更...             | 3-27 | 位置 .....                      | 7-12       |
| メッセージの置き換え .....            | 3-12 | テーブル・コンバータの位置                 | 7-12       |
| <b>get_wch</b> 関数 .....     | 4-14 | 別名ファイル .....                  | 7-12       |
| <b>get_wstr</b> 関数 .....    | 4-12 | <b>IL</b> エスケープ・シーケンス .....   | C-5        |
| <b>getchar</b> 関数 .....     | A-9  | <b>in_wchnstr</b> 関数 .....    | 4-10       |
| マルチバイト文字 .....              | 2-11 | <b>in_wchstr</b> 関数 .....     | 4-10       |
| <b>getch</b> 関数 .....       | 4-14 | <b>in_wch</b> 関数 .....        | 4-9        |
| <b>getc</b> 関数 .....        | A-9  | <b>innwstr</b> マクロ .....      | 4-11       |
| 制限事項 .....                  | 2-13 | <b>ins_nwstr</b> 関数 .....     | 4-7        |
| <b>getn_wstr</b> 関数 .....   | 4-12 | <b>ins_wch</b> 関数 .....       | 4-3        |
| <b>getnwstr</b> マクロ .....   | 4-12 | <b>ins_wstr</b> 関数 .....      | 4-7        |
| <b>gets</b> 関数 .....        | A-9  | <b>insnwstr</b> マクロ .....     | 4-7        |
| 制限事項 .....                  | 2-13 | <b>inswch</b> マクロ .....       | 4-3        |
| <b>gettext</b> 関数 .....     | 3-41 | <b>inswstr</b> マクロ .....      | 4-7        |
| <b>getwchar</b> 関数 .....    | A-9  | <b>inwchnstr</b> マクロ .....    | 4-10       |
| <b>getwch</b> 関数 .....      | 4-14 | <b>inwchstr</b> マクロ .....     | 4-10       |
| <b>getwc</b> 関数 .....       | A-9  | <b>inwch</b> マクロ .....        | 4-9        |
| メソッドの作成 .....               | 6-33 | <b>inwstr</b> マクロ .....       | 4-11       |
| <b>getwstr</b> マクロ .....    | 4-12 | <b>IRM</b> エスケープ・シーケンス ...    | C-5        |
| <b>GSM</b> エスケープ・シーケンス ..   | C-5  | <b>iscntrl</b> 関数 .....       | A-1        |
| <b>I</b>                    |      | <b>isdigit</b> 関数 .....       | A-1        |
| <b>I18N</b> .....           | 1-1  | <b>isgraph</b> 関数 .....       | A-1        |
| ( 国際化 も参照 )                 |      | <b>islower</b> 関数 .....       | A-1        |
| <b>ICH</b> エスケープ・シーケンス ...  | C-5  | <b>ISO C</b> 関数               |            |
| <b>iconv_close</b> 関数 ..... | A-12 | WPI 拡張 .....                  | A-6        |
|                             |      | <b>ISO/IEC 10646</b> 標準 ..... | 2-8        |
|                             |      | <b>ISO8859-15</b>             |            |

|                           |      |
|---------------------------|------|
| ユーロのサポート .....            | 2-9  |
| <b>ISO</b> コードセット .....   | 2-5  |
| <b>isprint</b> 関数 .....   | A-1  |
| <b>ispunct</b> 関数 .....   | A-1  |
| <b>isspace</b> 関数 .....   | A-1  |
| <b>isupper</b> 関数 .....   | A-1  |
| <b>iswalnum</b> 関数 .....  | A-1  |
| <b>iswalpha</b> 関数 .....  | A-1  |
| <b>iswcntrl</b> 関数 .....  | A-1  |
| <b>iswctype</b> 関数 .....  | A-2  |
| 文字クラスのテスト .....           | 6-13 |
| <b>iswdigit</b> 関数 .....  | A-1  |
| <b>iswgraph</b> 関数 .....  | A-1  |
| <b>iswlower</b> 関数 .....  | A-1  |
| <b>iswprint</b> 関数 .....  | A-1  |
| <b>iswpunct</b> 関数 .....  | A-1  |
| <b>iswspace</b> 関数 .....  | A-1  |
| <b>iswupper</b> 関数 .....  | A-1  |
| <b>iswxdigit</b> 関数 ..... | A-1  |

## L

|                                  |      |
|----------------------------------|------|
| <b>L10N (localization)</b> ..... | 1-2  |
| <b>langinfo</b> データベース .....     | 1-4  |
| strftime 関数 .....                | 2-21 |
| wcsftime 関数 .....                | 2-21 |
| 照会 .....                         | 2-20 |
| 情報 .....                         | 2-20 |
| メッセージ・カタログとの相違<br>点 .....        | 3-1  |
| <b>LANG</b> 環境変数                 |      |
| man コマンドの検索パス ....               | 7-10 |
| NLSPATH 設定 .....                 | 3-37 |
| setlocale の影響 .....              | 2-29 |

|                              |      |
|------------------------------|------|
| 検索パス内の %L .....              | 2-29 |
| メッセージ・カタログの生成                | 3-23 |
| ロケール・ファイル名のサフィックスを含む .....   | 2-29 |
| <b>Latin Cyrillic</b> コードセット |      |
| 言語サポート .....                 | 2-6  |
| <b>Latin Greek</b> コードセット    |      |
| 言語サポート .....                 | 2-6  |
| <b>Latin Hebrew</b> コードセット   |      |
| 言語サポート .....                 | 2-6  |
| <b>Latin-1</b> コードセット        |      |
| 言語サポート .....                 | 2-5  |
| <b>Latin-1</b> ロケール          |      |
| ユーロ以外の通貨記号 .....             | 6-24 |
| <b>Latin-2</b> コードセット        |      |
| 言語サポート .....                 | 2-5  |
| <b>Latin-4</b> コードセット        |      |
| 言語サポート .....                 | 2-5  |
| <b>Latin-5</b> コードセット        |      |
| 言語サポート .....                 | 2-6  |
| <b>Latin-9</b> コードセット        |      |
| 言語サポート .....                 | 2-6  |
| <b>LC_COLLATE</b>            |      |
| 照合重みの割り当て .....              | 6-15 |
| ロケール・ソース・ファイルでの定義 .....      | 6-14 |
| <b>LC_CTYPE</b>              |      |
| alnum 文字クラス .....            | 6-12 |
| 大文字/小文字変換 .....              | 6-13 |
| その他のオプション .....              | 6-13 |
| 定義されているクラス .....             | A-3  |
| 文字クラス .....                  | 6-11 |
| 文字クラス・キーワード .....            | 6-12 |
| 文字の範囲の指定 .....               | 6-12 |

|                             |      |                              |      |
|-----------------------------|------|------------------------------|------|
| ロケール・ソース・ファイルでの定義 .....     | 6-9  | methods ファイル .....           | 6-63 |
| <b>LC_MESSAGES</b>          |      | methods ファイルのコンパイル .....     | 6-65 |
| copy 文の使用 .....             | 6-21 | m オプション .....                | 6-65 |
| NLSPATH 設定 .....            | 3-37 | UNDEFINED 照合未指定 ....         | 6-17 |
| setlocale 関数による使用 .....     | 2-25 | v オプション .....                | 6-65 |
| 肯定応答 .....                  | 6-20 | w オプション .....                | 6-65 |
| 肯定応答の文字列の定義 .....           | 6-21 | シェアード・ライブラリの作成 .....         | 6-65 |
| 否定応答 .....                  | 6-20 | 詳細モード .....                  | 6-65 |
| 否定応答文字列 .....               | 6-21 | 省略時のメソッド .....               | 6-61 |
| ロケール・ソース・ファイルでの定義 .....     | 6-19 | シンボル値のインクリメント .              | 6-6  |
| <b>LC_MONETARY</b>          |      | ロケールの作成 .....                | 6-64 |
| copy 文 .....                | 6-24 | ロケールの作成時に使用されるファイル .....     | 6-1  |
| 定義可能なシンボル名 .....            | 6-22 | <b>locale</b> コマンド           |      |
| ユーロ以外の通貨 .....              | 6-24 | ロケール情報の表示 .....              | 3-33 |
| ユーロ文字 .....                 | 6-24 | <b>localtime</b> 関数          |      |
| ロケール・ソース・ファイルでの定義 .....     | 6-21 | strftime 関数 .....            | 2-21 |
| <b>LC_NUMERIC</b>           |      | <b>LOCPATH</b> 環境変数 .....    | 6-65 |
| ロケール・ソース・ファイルでの定義 .....     | 6-24 | iconv コマンドの影響 .....          | 7-12 |
| <b>LC_TIME</b>              |      | <b>M</b>                     |      |
| copy 文 .....                | 6-28 | <b>manpage</b>               |      |
| ロケールでの定義 .....              | 6-25 | ( リファレンス・ページ を参照 )           |      |
| <b>ld</b> コマンド              |      | <b>man</b> コマンド .....        | 7-10 |
| ロケール・メソッド・ライブラリの構築 .....    | 6-62 | リファレンス・ページの翻訳                | 7-10 |
| <b>libiconv</b> ライブラリ ..... | A-12 | <b>mblen</b> 関数 .....        | A-7  |
| <b>localeconv</b> 関数 .....  | A-5  | メソッドの作成 .....                | 6-40 |
| 数値の書式付け .....               | 2-22 | <b>mbrlen</b> 関数 .....       | A-7  |
| <b>localedef</b> コマンド       |      | <b>mbrtowc</b> 関数 .....      | A-7  |
| cv オプション .....              | 6-62 | <b>mbsinit</b> 関数 .....      | A-9  |
| f オプション .....               | 6-65 | <b>mbsrtowcs</b> 関数 .....    | A-7  |
| i オプション .....               | 6-65 | <b>__mbstopcs</b> メソッド ..... | 6-30 |

|                             |            |                                        |      |
|-----------------------------|------------|----------------------------------------|------|
| <b>mbstowcs</b> 関数 .....    | 2-14, A-7  | <b>XtSetLanguageProc</b> 呼び出しの要件 ..... | 5-5  |
| メソッドの作成 .....               | 6-44       | グリフの編集 .....                           | B-10 |
| <b>__mbtopc</b> メソッド .....  | 6-33       | 言語設定についての注意 .....                      | 5-6  |
| <b>mbtowc</b> 関数 .....      | 2-12, A-7  | 言語の設定 .....                            | 5-5  |
| メソッドの作成 .....               | 6-46       | コンパウンド・ストリング ....                      | 5-8  |
| <b>mkcatdefs</b> コマンド ..... | 3-23       | テキスト・ウィジェットの利用 .....                   | 5-6  |
| delset ディレクティブ .....        | 3-9        | テキスト翻訳の問題 .....                        | 3-14 |
| makefile での利用 .....         | 3-23       | フォントセットの利用 .....                       | 5-6  |
| 一般的なエラー .....               | 3-26       | メッセージ・カタログ .....                       | 1-4  |
| 識別子と番号のマッピング ...            | 3-25       | メッセージの処理 .....                         | 3-1  |
| シンボリック名の数値への変換 .....        | 3-10       | 両方向へのテキスト表示 .....                      | 5-8  |
| 制限事項とガイドライン ....            | 3-25       | ロケールの設定 .....                          | 5-6  |
| セットからのすべてのメッセージの削除 .....    | 3-12       | ロード可能フォントの要件 ..                        | B-26 |
| セット識別子の指定時 .....            | 3-8        | <b>Motif</b> インタフェース                   |      |
| 対話型の利用 .....                | 3-22       | コードとメッセージの分離 ....                      | 1-3  |
| 不完全なメッセージ・ヘッダ・ファイル .....    | 3-25       | <b>mvadd_wchstr</b> 関数 .....           | 4-4  |
| 複数のソース・ファイルの処理 .....        | 3-30       | <b>mvadd_wch</b> 関数 .....              | 4-2  |
| ヘッダ・ファイルの作成 ....            | 3-22       | <b>mvaddnwstr</b> マクロ .....            | 4-5  |
| ポータビリティ .....               | 3-25       | <b>mvaddw_wchnstr</b> 関数 .....         | 4-4  |
| 無視される行 .....                | 3-13       | <b>mvaddwchnstr</b> マクロ .....          | 4-4  |
| メッセージ・テキスト・ソースの処理 .....     | 3-22       | <b>mvaddwchstr</b> マクロ .....           | 4-4  |
| メッセージの置き換え .....            | 3-12       | <b>mvaddwch</b> マクロ .....              | 4-2  |
| メッセージの削除 .....              | 3-11       | <b>mvaddwstr</b> マクロ .....             | 4-5  |
| <b>mkfontdir</b> コマンド ..... | B-25       | <b>mvdelch</b> マクロ .....               | 4-8  |
| <b>MNLS</b> .....           | 4-1        | <b>mvget_wch</b> 関数 .....              | 4-14 |
| <b>Motif</b> アプリケーション ..... | 5-5        | <b>mvget_wstr</b> 関数 .....             | 4-12 |
| UDC フォントの作成 .....           | B-10, B-21 | <b>mvgetch</b> 関数 .....                | 4-14 |
|                             |            | <b>mvgetn_wstr</b> 関数 .....            | 4-12 |
|                             |            | <b>mvgetnwstr</b> マクロ .....            | 4-12 |
|                             |            | <b>mvgetwch</b> 関数 .....               | 4-14 |
|                             |            | <b>mvgetwstr</b> マクロ .....             | 4-12 |
|                             |            | <b>mvin_wchnstr</b> 関数 .....           | 4-10 |

**mvinnwstr** マクロ..... 4-11  
**mvins\_nwstr** 関数..... 4-7  
**mvins\_wch** 関数..... 4-3  
**mvins\_wstr** 関数..... 4-7  
**mvinsnwstr** マクロ..... 4-7  
**mvinswch** マクロ..... 4-3  
**mvinswstr** マクロ..... 4-7  
**mvinwchnstr** マクロ..... 4-10  
**mvinwchstr** マクロ..... 4-10  
**mvinwch** マクロ..... 4-9  
**mvinwstr** マクロ..... 4-11  
**mvprintw** 関数..... 4-16  
**mvscanw** 関数..... 4-15  
**mvw\_getwch** 関数..... 4-14  
**mvwadd\_wchnstr** 関数..... 4-4  
**mvwadd\_wchstr** 関数..... 4-4  
**mvwadd\_wch** 関数..... 4-2  
**mvwaddnwstr** マクロ..... 4-5  
**mvwaddwchnstr** マクロ..... 4-4  
**mvwaddwchstr** マクロ..... 4-4  
**mvwaddwch** マクロ..... 4-2  
**mvwaddwstr** マクロ..... 4-5  
**mvwdelch** 関数..... 4-8  
**mvwdelch** マクロ..... 4-8  
**mvwget\_wstr** 関数..... 4-12  
**mvwgetch** 関数..... 4-14  
**mvwgetn\_wstr** 関数..... 4-12  
**mvwgetnwstr** マクロ..... 4-12  
**mvwgetwch** 関数..... 4-14  
**mvwgetwstr** マクロ..... 4-12  
**mvwin\_wchnstr** 関数..... 4-10  
**mvwin\_wchstr** 関数..... 4-10  
**mvwin\_wch** 関数..... 4-9

**mvwinnwstr** マクロ..... 4-11  
**mvwins\_nwstr** 関数..... 4-7  
**mvwins\_wstr** 関数..... 4-7  
**mvwinsnwstr** マクロ..... 4-7  
**mvwinswch** マクロ..... 4-3  
**mvwinswstr** マクロ..... 4-7  
**mvwinwchnstr** マクロ..... 4-10  
**mvwinwchstr** マクロ..... 4-10  
**mvwinwch** マクロ..... 4-9  
**mvwinwstr** マクロ..... 4-11  
**mvwprintw** 関数..... 4-16  
**mvwscanw** 関数..... 4-15

## N

### neqn プリプロセッサ

tbl コマンドと nroff コマンド 7-10

**NL\_CAT\_LOCALE** 定数..... 2-25

**nl\_catd** 型..... 3-35

プログラムでの宣言..... 2-24

**nl\_langinfo** 関数..... A-5

CODESET で返される値..... 6-4

langinfo データベース..... 2-20

strftime 関数の引数..... 2-22

**NL\_MSGMAX** 定数..... 3-10

**NL\_SETD** 定数..... 2-31

省略時のメッセージ・セット値の定義..... 3-8

**NL\_SETMAX** 定数..... 3-8

**NL\_TEXTMAX** 定数

メッセージ・テキスト・パラメー

タ..... 3-11

**NLSPATH** 環境変数..... 3-35

catclose 関数による使用..... 2-25

catopen 関数による使用..... 2-25,  
3-35

catopen は無視..... 3-39

LC\_MESSAGES 設定..... 3-37

設定の置換フィールド..... 3-36

**NLSPATH** 設定の置換フィールド..... 3-36

**noexpr** キーワード..... 6-20, 6-21

**nostr** キーワード..... 6-21

**nroff** コマンド..... 7-7

.ki..... 7-8

.kl..... 7-9

.ko..... 7-8

neqn 数式フォーマット 7-10

後にスペースを置くことが可能 7-9

行の折り返し..... 7-8

行の折り返しの規則..... 7-8

テキスト揃えの規則..... 7-9

表意文字..... 7-9

**nroff** によるテキスト揃え..... 7-9

## O

**Off-the-Spot** 前編集スタイル

自動サイズ変更の要件..... 5-7

テキスト・ウィジェット..... 5-7

入力システム..... 7-2

**On-the-Spot** 前編集スタイル

Cb オプション..... 5-27

XIC オブジェクト作成の要件 5-30

コールバックの要件..... 5-31

テキスト・ウィジェット..... 5-7

入力システム..... 7-2

**order\_start** キーワード..... 6-15

**Over-the-Spot** 前編集スタイル

テキスト・ウィジェット..... 5-7

入力システム..... 7-2

## P

**PCS**..... 2-12

文字の代用..... 1-7

利用可能な文字..... 1-7

--**pcstombs** メソッド..... 6-38

--**pctomb** メソッド..... 6-40

**PostScript** フォント

フォント・レンダラ..... 7-14

**printf** 関数..... A-6

catgets 関数..... 2-26

X ウィンドウ・アプリケーション

ン..... 5-11

書式指定子..... 2-26

制限事項..... 2-13

**printf** コマンド

書式付き出力の書き込み..... 3-33

**printw** 関数..... 4-16

**putc** 関数..... A-9

制限事項..... 2-13

**puts** 関数..... A-9

制限事項..... 2-13

**putwc** 関数..... A-9

## Q

**quote** ディレクティブ

ソース・ファイル内に複数... 3-13

## R

### Reset Terminal オプション

DECterm の罫線..... C-5

**RIS** エスケープ・シーケンス.... C-5

### Root Window 前編集スタイル

テキスト・ウィジェット ..... 5-7

入力システム ..... 7-3

## S

**scanf** 関数 ..... A-6

書式指定子..... 2-27

制限事項 ..... 2-13

**scanw** 関数 ..... 4-15

**setlocale** 関数 ..... A-1

category 引数 ..... 2-28

locale\_name 引数 ..... 2-29

X アプリケーション..... 5-9

事前設定されているロケールへのバ  
インド..... 2-29

特定のロケール・カテゴリの変

更 ..... 2-31

ロケール設定の変更..... 2-30

ロケールの初期化 ..... 2-28

**set** ディレクティブ..... 3-8

メッセージ・セットの削除... 3-11

**SoftODL** サービス..... B-1, B-20

**sort** コマンド ..... 7-6, B-20  
( **asort** コマンド も参照 )

**sprintf** 関数 ..... A-6

**sscanf** 関数 ..... A-6

**strcat** 関数..... A-10

**strchr** 関数 ..... A-10

**strcmp** 関数 ..... A-11

制限事項 ..... 2-18

**strcoll** 関数 ..... A-5

制限事項..... 2-18

利点 ..... 2-18

**strcpy** 関数 ..... A-11

**strcspn** 関数..... A-10

**strextract** コマンド ..... 3-18

作成されるファイル..... 3-18

パターン・ファイル..... 3-18

無視ファイル ..... 3-19

**strfmon** 関数 ..... A-5

金額値の書式付け..... 2-22

**strftime** 関数 ..... A-5

langinfo データベース ..... 2-21

time 関数と localtime 関数 ... 2-21

引数としての nl\_langinfo 関数 2-22

日付と時刻の書式付け ..... 2-21

日付または時刻への変換 ..... 2-22

**strlen** 関数..... A-11

**strmerge** コマンド ..... 3-18

作成されるファイル..... 3-18

**strncat** 関数 ..... A-10

**strncmp** 関数..... A-11

**strncpy** 関数..... A-11

**strpbrk** 関数..... A-10

**strptime** 関数 ..... A-5

**strrchr** 関数 ..... A-10

**strstr** 関数 ..... A-10

**strtod** 関数 ..... A-7

**strtok** 関数 ..... A-11

**strtol** 関数 ..... A-7

**strtoul** 関数..... A-7

**stty** コマンド

odl オプション ..... B-1

**swprintf** 関数 ..... A-6

**swscanf** 関数 ..... A-6



**System V Multi-National  
Language Supplement**

curses ライブラリ..... 4-1

## T

**tbl** コマンド..... 7-10

neqn 数式フォーマット ..... 7-10

**time** 関数

strftime 関数..... 2-21

**tolower** 関数 ..... 6-13, A-4

**toupper** 関数 ..... 6-13, A-4

**towctrans** 関数..... A-4

**towlower** 関数..... 6-13, A-4

利点 ..... 2-17

**towupper** 関数 ..... 6-13, A-4

利点 ..... 2-17

**trans** コマンド

メッセージ・カタログの翻訳 3-21

**trans** ユーティリティ

メッセージ・ファイル内のテキスト  
の検索..... 3-13

**TrueType** フォント..... 7-16

## U

**UCS**..... 1-8, 2-8

**UCS-2**..... 1-8, 2-8

**UCS-4**

コードセット ..... 2-8

サポート..... 1-10

**UCS-4** 処理コード

UTF-8 データの変換 ..... 2-9

**UDC**

cedit コマンド ..... B-16

ULTRIX からの変換 ..... B-4

アジア系言語 ..... 7-4

アジア系言語の制限事項 ..... B-8

クラスの作成 ..... B-7

言語サポート ..... B-4

言語とコードセットの設定..... B-8

コードセット値の作成 ..... B-7

削除 ..... B-7

作成 ..... B-1, B-3

照合重み..... 7-6

属性 ..... B-3

名前の作成..... B-7

入力キー・シーケンスの作成 . B-7

表示のセットアップ..... B-1

ファイルのオンデマンド・ローディ  
ング ..... B-1

フォント・グリフの作成 ..... B-9

フォント・サイズの選択 ..... B-10

フォントのスケーリング ..... B-8

文字属性レコード..... B-3, B-7

**UDC** エディタ ..... B-3

文字属性..... B-3

**UDC** データベース..... 7-4

位置構成の cp\_dirs ファイル .. 7-4

オン・デマンド・ローディング B-1

個人用 ..... B-3

サポート・ファイル..... B-20

システム全体 ..... B-3

省略時の位置の設定..... 7-4

省略時のパス ..... B-4

フォント・ファイル..... B-20

フォント・レンダラ..... 7-15

|                                 |          |
|---------------------------------|----------|
| <b>UDC フォント</b>                 |          |
| bdf フォーマット.....                 | B-21     |
| pcf フォーマット.....                 | B-21     |
| 標準のフォントとのマージ..                  | B-27     |
| <b>UDC フォント・コンバータ</b>           |          |
| Motif アプリケーション用...              | B-12     |
| <b>UDC 文字</b>                   |          |
| 文字編集のコード.....                   | B-9      |
| <b>UID ファイル</b>                 |          |
| CDE .....                       | 5-2      |
| <b>UNDEFINED 文</b>              |          |
| オペランド.....                      | 6-17     |
| 利点 .....                        | 6-17     |
| <b>ungetch 関数</b> .....         | 4-14     |
| <b>ungetc 関数</b> .....          | A-9      |
| <b>ungetwch 関数</b> .....        | 4-14     |
| <b>ungetwc 関数</b> .....         | A-9      |
| <b>Unicode</b> .....            | 1-8, 2-8 |
| ( UCS も参照 )                     |          |
| 標準 .....                        | 2-8      |
| <b>Unicode ロケール</b> .....       | 2-2      |
| dense コードと同じ charmap.           | 6-19     |
| dense コードとの類似点 .....            | 2-4      |
| 標準 .....                        | 2-3      |
| プライベート用領域.....                  | 2-4      |
| ワイド文字・エンコーディング                  | 2-3      |
| <b>Universal Transformation</b> |          |
| Format .....                    | 1-8      |
| ( UTF も参照 )                     |          |
| <b>universal.UTF-8</b>          |          |
| 使用する場合 .....                    | 2-9      |
| <b>UNIX 標準</b> .....            | 1-1      |
| <b>UTF</b>                      |          |
| システム上でサポートされている                 |          |
| フォーマット .....                    | 1-8      |
| 推奨 .....                        | 1-8      |
| <b>UTF-16</b>                   |          |
| UCS-2 .....                     | 1-8      |
| UCS-4 .....                     | 1-8      |
| 代替文字.....                       | 1-8      |
| バイト・オーダ .....                   | 1-9      |
| <b>UTF-32</b>                   |          |
| 制約 .....                        | 1-9      |
| 内部処理コード .....                   | 2-3      |
| バイト・オーダ .....                   | 1-9      |
| <b>UTF-32 処理コード</b>             |          |
| ロケールの一覧 .....                   | 2-9      |
| <b>UTF-8</b> .....              | 2-8      |
| UCS-4 エンコーディング.....             | 1-8      |
| UCS-4 への変換.....                 | 2-9      |
| コンバータとロケール .....                | 1-8      |
| ユーロのサポート.....                   | 2-9      |
| <b>UTF-8 ロケール</b> .....         | 2-3      |
| universal .....                 | 2-3      |
| マルチバイト・データ .....                | 2-3      |
| <b>V</b>                        |          |
| <b>vfprintf 関数</b> .....        | A-6      |
| <b>vfwprintf 関数</b> .....       | A-6      |
| <b>vprintf 関数</b> .....         | A-6      |
| <b>vsprintf 関数</b> .....        | A-6      |
| <b>vswprintf 関数</b> .....       | A-6      |
| <b>vw_printw 関数</b> .....       | 4-16     |
| <b>vw_scanw 関数</b> .....        | 4-15     |
| <b>vwprintf 関数</b> .....        | A-6      |
| <b>vwprintw 関数</b> .....        | 4-16     |
| <b>vwscanw 関数</b> .....         | 4-15     |

## W

- wadd\_wchnstr** 関数..... 4-4
- wadd\_wchstr** 関数..... 4-4
- wadd\_wch** 関数..... 4-2
- waddnwstr** 関数..... 4-5
- waddwchnstr** 関数..... 4-4
- waddwchstr** マクロ..... 4-4
- waddwch** 関数..... 4-2
- waddwstr** マクロ..... 4-5
- wchar\_t**
  - ヘッダ・ファイルの説明..... 1-6
- wcrtomb** 関数..... A-7
- wcscat** 関数..... A-10
- wcschr** 関数..... A-10
- wcscmp** 関数..... A-11
  - 制限事項..... 2-18
- wcscoll** 関数..... A-5
  - 利点..... 2-18
- wcscpy** 関数..... A-11
- wcscspn** 関数..... A-10
- wcsftime** 関数..... A-5
  - langinfo データベース..... 2-21
- wcslen** 関数..... A-11
- wcsncat** 関数..... A-10
- wcsncmp** 関数..... A-11
- wcsncpy** 関数..... A-11
- wcspbrk** 関数..... A-10
- wcsrchr** 関数..... A-10
- wcsrtombs** 関数..... A-7
- wcsstr** 関数..... A-10
- wctod** 関数..... A-7
- wctok** 関数..... A-11
- wctol** 関数..... A-7
- wcstombs** 関数..... A-7
  - メソッドの作成..... 6-50
- wcstoul** 関数..... A-7
- wcswcs** 関数..... A-10
- wcswidth** 関数..... A-12
  - メソッドの作成..... 6-56
- wcsxfrm** 関数
  - 利点..... 2-18
- wctomb** 関数..... A-7
  - メソッドの作成..... 6-53
- wctrans** 関数..... A-4
- wctype** 関数..... A-2
  - 文字クラスのテスト..... 6-13
- wcwidth** 関数..... A-12
  - メソッドの作成..... 6-59
- wecho\_wchar** 関数..... 4-2
- wechowchar** マクロ..... 4-2
- wget\_wch** 関数..... 4-14
- wget\_wstr** 関数..... 4-12
- wgetch** 関数..... 4-14
- wgetn\_wstr** 関数..... 4-12
- wgetnwstr** 関数..... 4-12
- wgetwch** 関数..... 4-14
- wgetwstr** マクロ..... 4-12
- win\_wchnstr** 関数..... 4-10
- win\_wchstr** 関数..... 4-10
- win\_wch** 関数..... 4-9
- winnwstr** 関数..... 4-11
- wins\_nwstr** 関数..... 4-7
- wins\_wch** 関数..... 4-3
- wins\_wstr** 関数..... 4-7
- winsnwstr** 関数..... 4-7
- winswch** 関数..... 4-3

**winswstr** マクロ ..... 4-7  
**winwnstr** 関数 ..... 4-10  
**winwchstr** マクロ ..... 4-10  
**winwch** 関数 ..... 4-9  
**winwstr** マクロ ..... 4-11  
**WLS** サブセット  
     提供されるロケール ..... 2-5  
**wmemchr** 関数 ..... A-12  
**wmemcmp** 関数 ..... A-12  
**wmemcpy** 関数 ..... A-12  
**wmemmove** 関数 ..... A-12  
**wmemset** 関数 ..... A-12  
**WPI**  
     langinfo データの取得 ..... A-5  
     インタフェースの一覧 ..... A-1  
     大文字/小文字変換関数 ..... A-4  
     書き込み関数 ..... A-6  
     コードセット変換関数 ..... A-12  
     数値変換関数 ..... A-7  
     入出力関数 ..... A-9  
     日付および時刻の値の書式付け ..... A-5  
     マルチバイト文字とワイド文字の変換 ..... A-7  
     文字照合関数 ..... A-5  
     文字分類関数 ..... A-1  
     文字列処理関数 ..... A-10  
     読み取り関数 ..... A-6  
     ロケール宣言関数 ..... A-1  
**WPI** インタフェース  
     テキストを渡す ..... 5-11  
**WPI** 拡張  
     ISO C 関数 ..... A-6  
**wprintf** 関数 ..... A-6  
**wprintw** 関数 ..... 4-16  
**wscanf** 関数 ..... A-6

**wscanw** 関数 ..... 4-15  
**mvwins\_wch** 関数 ..... 4-3

## X

**X/Open** 標準  
     コードセットの要件 ..... 2-15  
     メッセージ・システムの要件 ..... 2-24  
**X11R6** ..... 5-1  
**XBaseFontNameListOfFontSet** 関数 ..... 5-13  
**XCloseIM** 関数 ..... 5-24, 5-25  
**XCloseOM** 関数 ..... 5-18  
**XCreateFontSet** 関数 ..... 5-13  
**XCreateIC** 関数 ..... 5-28  
     失敗の条件 ..... 5-30  
**XCreateOC** 関数 ..... 5-18  
**XDefaultString** 関数 ..... 5-20  
**XDestroyOC** 関数 ..... 5-18  
**XDestroy** 関数 ..... 5-28  
**XDisplayOfIM** 関数 ..... 5-24  
**XDisplayOfOM** 関数 ..... 5-18  
**XDm** ライブラリ ..... 5-5  
**XDrawImageString16** 関数 ..... 5-15  
**XDrawString16** 関数 ..... 5-15  
**XDrawString** 関数 ..... 5-15  
**XDrawText16** 関数 ..... 5-15  
**XDrawText** 関数 ..... 5-15  
**XExtentsOfFontSet** 関数 ..... 5-15  
**XFillRectangle** 関数 ..... 5-17  
**XFilterEvent** 関数 ..... 5-34  
     XtDispatchEvent 関数による呼び出し ..... 5-35  
**XFontSetExtents** 構造体 ..... 5-15  
**XFontSet** オブジェクト ..... 5-10

|                                  |            |                                    |            |
|----------------------------------|------------|------------------------------------|------------|
| <b>XFontSet</b> 構造体 .....        | 5-12       | <b>XmbDrawText</b> 関数 .....        | 5-15       |
| Xt ルーチンのサポート .....               | 5-4        | <b>XmbLookupString</b> 関数 .....    | 5-35, 5-37 |
| リソース属性 .....                     | 5-4        | <b>XmbResetIC</b> 関数 .....         | 5-30       |
| <b>XFontsOfFontSet</b> 関数 .....  | 5-13       | <b>XmbSetWMProperties</b> 関数 ..    | 5-20, 5-22 |
| <b>XFontStruct</b> 構造体 .....     | 5-12       | <b>XmbTextEscapement</b> 関数 ..     | 5-15, 5-18 |
| <b>XFreeFontSet</b> 関数 .....     | 5-13       | <b>XmbTextExtents</b> 関数 .....     | 5-15       |
| <b>XGetICValues</b> 関数 .....     | 5-30       | <b>XmbTextListToTextProperty</b> 関 | 数 .....    |
| XNFilterEvents 引数 .....          | 5-34       | 数 .....                            | 5-20       |
| <b>XGetIMValues</b> 関数 .....     | 5-25, 5-27 | <b>XmbTextPerCharExtents</b> 関     | 数 .....    |
| <b>XGetOCValues</b> 関数 .....     | 5-18       | 数 .....                            | 5-15       |
| <b>XGetOMValues</b> 関数 .....     | 5-18       | <b>XmbTextPropertyToTextList</b> 関 | 数 .....    |
| <b>XIC</b> オブジェクト .....          | 5-10, 5-23 | 数 .....                            | 5-20       |
| XNClientWindow 属性 .....          | 5-30       | <b>XMODIFIERS</b> 環境変数 .....       | 5-10       |
| 管理 .....                         | 5-30       | <b>XmStringCreateLocalized</b> 関   | 数 .....    |
| 作成と使用 .....                      | 5-28       | 数 .....                            | 5-8        |
| 属性 .....                         | 5-28       | <b>XmStringCreate</b> 関数 .....     | 5-8        |
| 属性の指定 .....                      | 5-30       | <b>XmTextField</b> ウィジェット .....    | 5-7        |
| 破棄 .....                         | 5-28       | フォント検索パターン .....                   | 5-6        |
| 前編集コールバックの登録 ..                  | 5-30       | <b>XmText</b> ウィジェット .....         | 5-7        |
| 明示的なクローズ .....                   | 5-39       | フォント検索パターン .....                   | 5-6        |
| <b>ximdemo</b> アプリケーション .....    | 5-9        | <b>Xm</b> ライブラリ .....              | 5-5        |
| <b>XIMOfIC</b> 関数 .....          | 5-30       | <b>XNDestroyCallback</b> リソース      | 5-37       |
| <b>XIM</b> オブジェクト .....          | 5-10, 5-23 | <b>XNQueryInputStyle</b> 関数 .....  | 5-25       |
| IM サーバが失敗した場合のクロー                |            | <b>XOC</b> オブジェクト .....            | 5-10       |
| ズ .....                          | 5-37       | ロケール依存のテキストの表                      |            |
| 接続と接続の解除 .....                   | 5-23       | 示 .....                            | 5-18       |
| <b>XLocaleOfFontSet</b> 関数 ..... | 5-13       | <b>XOMOfOC</b> 関数 .....            | 5-18       |
| <b>XLocaleOfIM</b> 関数 .....      | 5-24       | <b>XOM</b> オブジェクト .....            | 5-10       |
| <b>XLocaleOfOM</b> 関数 .....      | 5-18       | ロケール依存のテキストの表                      |            |
| <b>XLookupString</b> 関数 .....    | 5-37       | 示 .....                            | 5-18       |
| <b>XmbDrawImageString</b> 関数 ..  | 5-15, 5-17 |                                    |            |
| <b>XmbDrawString</b> 関数 .....    | 5-15       |                                    |            |

**XOpenIM** 関数 ..... 5-23, 5-25  
     失敗時の省略時の動作 ..... 5-24  
     失敗の条件..... 5-24  
**XOpenOM** 関数..... 5-18  
**xpg4demo**  
     サンプル・アプリケーション . 2-1  
**XResourceManagerString** 関  
     数..... 5-22  
**XrmDatabase** コンポーネント. 5-10  
**XrmGetFileDatabase** 関数 ... 5-22  
**XrmGetStringDatabase** 関数. 5-22  
**XrmLocaleOfDatabase** 関数.. 5-22  
**XrmPutFileDatabase** 関数 ... 5-22  
**XrmPutLineResource** 関数... 5-22  
**XSelectInput** 関数 ..... 5-34  
**XSetICFocus** 関数..... 5-30, 5-37  
**XSetICValues** 関数 ..... 5-30  
**XSetIMValues** 関数 ..... 5-37  
**XSetLocaleModifiers** 関数.... 5-3,  
     5-10  
**XSetOCValues** 関数 ..... 5-18  
**XSetOMValues** 関数..... 5-18  
**xset** コマンド ..... B-25  
**XSH CAE** 仕様  
     含まれている関数..... A-1  
**XSupportsLocale** 関数.. 5-3, 5-10  
**XtAppInitialize** 関数..... 5-4  
**XtDispatchEvent** 関数.. 5-4, 5-35  
**XtDisplayInitialize** 関数..... 5-4  
**XtInitialize** 関数 ..... 5-4  
**XtOpenDisplay** 関数 ..... 5-4  
**XtSetLanguageProc** 関数..... 5-3  
**XtSetLanguageProc** 呼び出し  
     Motif 国際化に必要 ..... 5-5  
**Xt** ライブラリ  
     国際化機能..... 5-3  
     異なるリリースでの国際化.... 5-5  
     コードセット ..... 5-5  
     入力サーバ..... 5-4  
     フォントセット ..... 5-4  
     ロケールとリソースのパラドク  
         ス ..... 5-3  
     ロケールの設定 ..... 5-3  
**Xt** ルーチン  
     XtSetLanguageProc 呼び出しの要  
         件 ..... 5-3  
**XUnsetICFocus** 関数... 5-30, 5-37  
**XVaCreateNestedList** 関数 ... 5-29  
**XwcDrawImageString** 関数.. 5-15  
**XwcDrawString** 関数 ..... 5-15  
**XwcDrawText** 関数 ..... 5-15  
**XwcFreeStringList** 関数..... 5-20  
**XwcLookupString** 関数..... 5-35  
**XwcResetIC** 関数 ..... 5-30  
**XwcTextEscapement** 関数.... 5-15  
**XwcTextExtents** 関数..... 5-15  
**XwcTextListToTextProperty** 関  
     数..... 5-20  
**XwcTextPerCharExtents** 関  
     数..... 5-15  
**XwcTextPropertyToTextList** 関  
     数..... 5-20  
**X** アプリケーション  
     UDC フォントの作成 ..... B-21  
     移植性のあるものの開発 ..... 5-25  
     イベント・フィルタリング... 5-34  
     多国語対応の開発..... 5-10  
     テキストのエンコードを処理する関  
         数 ..... 5-15  
     テキスト翻訳の問題..... 3-14

マルチバイト PostScript フォントの  
 使用 ..... 7-14  
 メッセージの処理 ..... 3-1  
 文字とキー・シンボルの取得 5-35  
 ロケールの設定 ..... 5-9  
 ロード可能フォントの要件.. B-26  
**X** ツールキット..... 5-3  
 (Xt ライブラリ も参照)  
**X** ツールキット・イントリンシク  
 ス..... 5-4  
 (Xt ライブラリ も参照)  
**X** ライブラリ  
 クライアント間通信のテキス  
 ト ..... 5-20  
 国際化機能の使用 ..... 5-1  
 入力サーバの使用 ..... 5-39  
 入力処理の要約 ..... 5-39

## Y

**yesexpr** キーワード..... 6-20  
**yesstr** キーワード..... 6-21  
**yesxpr** キーワード..... 6-21

## あ

アジア系言語  
 エスケープ・シーケンス ..... C-1  
 語句入力システム ..... 7-4  
 コードセット ..... 2-6  
 コードセットと X ウィンドウのフォ  
 ント ..... 5-12  
 照合 ..... 7-6  
 テキストのプリント..... 7-10

テクニカル・リファレンス.... 7-1  
 入力システム ..... 7-2  
 ユーザ定義文字 ..... 7-4  
 アジア系言語サポート..... 2-8  
 アジア系言語の文字  
 画数 ..... 1-5  
 部首 ..... 1-5  
 部首と画数によるソート ..... 1-5  
 値データベースとの照合  
 省略時の位置の設定..... 7-4  
 アプリケーション  
 dense コード・ロケール ..... 2-3  
 内部処理コード ..... 1-10  
 入力サーバの実行..... 7-3  
 複数言語のサポート..... 2-1  
 ロケールと処理コード ..... 2-2  
 ロケールとマルチバイト文字 . 2-2  
 ロケールの設計 ..... 2-3  
 アプリケーションのプログラミング  
 国際化での注意 ..... 7-1

## え

エスケープ文字  
 charmap ファイル ..... 6-4  
 nroff コマンドの設定..... 7-8  
 メッセージ文字列..... 3-6  
 ロケール・ソースでの再定義 . 6-8  
 ロケール・ソース・ファイルでの設  
 定 ..... 6-8  
 エンコーディング・フォーマット  
 UCS-2 ..... 2-8  
 UTF ..... 2-8

オペレーティング・システムでのサ  
ポート..... 2-8

## お

応答文字列  
ロケールでの定義..... 6-19  
大文字  
テスト..... 2-16  
大文字/小文字変換..... A-4  
大文字と小文字の変換.... 1-4, 2-17  
オペレーティング・システム  
国際化インタフェース..... 1-1  
国際化ユーティリティ..... 1-1

## か

改行文字  
メッセージ文字列..... 3-6  
改ページ文字  
メッセージ文字列..... 3-6  
カタログ  
メッセージの取り出し..... 2-24  
カッコ文字  
行の折り返し..... 7-8  
画面処理  
文字セル端末..... 4-1

## き

基準文字属性データベース..... B-4  
共通デスクトップ環境  
( CDE を参照 )  
行頭禁則文字..... 7-8  
個人用セットの定義..... 7-9  
行の折り返し

nroff コマンド..... 7-8  
行末禁則文字..... 7-8  
個人用セットの定義..... 7-9  
金額値  
書式..... 2-22  
キーボード  
コンパウンド・ストリングの取  
得..... 5-35  
サポートされていない文字の入  
力..... 2-15

## く

句読点文字  
行の折り返し..... 7-8  
グラフィック  
埋め込みテキスト..... 3-14

## け

桁のグループ・サイズ  
localeconv 関数による調査... 2-22  
言語  
構文の作成..... 2-26  
国際化ソフトウェア..... 1-3  
宣言..... 1-1  
文字の扱い..... 1-3  
言語サポート  
Latin Cyrillic コードセット... 2-6  
Latin Greek コードセット... 2-6  
Latin Hebrew コードセット... 2-6  
Latin-1 コードセット..... 2-5  
Latin-2 コードセット..... 2-5  
Latin-4 コードセット..... 2-5  
Latin-5 コードセット..... 2-6  
Latin-9 コードセット..... 2-6



|              |     |
|--------------|-----|
| 言語バリエーション    |     |
| ドキュメント ..... | 7-1 |

## こ

|                          |           |
|--------------------------|-----------|
| 肯定応答                     |           |
| ロケールでの定義 .....           | 6-19      |
| 国際化 .....                | 1-1       |
| 照合アルゴリズム .....           | 2-18      |
| 国際化関数 .....              | 2-1       |
| 国際化ソフトウェア                |           |
| 開発ツール .....              | 2-1       |
| 特徴 .....                 | 2-1       |
| 語句データベース                 |           |
| 省略時の位置の設定 .....          | 7-4       |
| 語句入力システム .....           | 7-4       |
| コメント                     |           |
| charmap ファイル .....       | 6-3       |
| 区切り文字の再定義 .....          | 6-8       |
| メッセージの set ディレクティブ ..... | 3-8       |
| ロケール・ソース・ファイル ..         | 6-8       |
| コメント文字                   |           |
| methods ファイル .....       | 6-64      |
| 小文字                      |           |
| テスト .....                | 2-16      |
| コンパウンド・ストリング             |           |
| Motif アプリケーションでの作成 ..... | 5-8       |
| コードセット .....             | 1-5, 2-8  |
| 8 進値の使用 .....            | 2-11      |
| ASCII .....              | 2-4, 2-11 |
| catopen 関数による変換 .....    | 3-39      |
| ISO .....                | 2-5       |

|                         |      |
|-------------------------|------|
| man コマンドの変換 .....       | 7-10 |
| X/Open 準拠の規則 .....      | 2-15 |
| アジア系言語のサポート .....       | 2-6  |
| 大文字と小文字の変換 .....        | 2-17 |
| 言語要件 .....              | 2-1  |
| 交換メディア .....            | 2-6  |
| コード内リテラル .....          | 2-11 |
| 最上位ビットの使用 .....         | 2-11 |
| 作成 .....                | 6-2  |
| 使用時の問題 .....            | 2-11 |
| 状態依存のエンコーディング ..        | 2-15 |
| シングルバイト文字 .....         | 2-11 |
| ソース・バージョンと実行バージョン ..... | 2-15 |
| ソース・バージョンの規則 ..         | 2-15 |
| データの透過性 .....           | 2-11 |
| データ・ファイルの変換 .....       | 7-11 |
| 名前の設定 .....             | 6-4  |
| ヌル文字 .....              | 2-15 |
| ネットワーク上での使用 .....       | 2-6  |
| ファイルのコードセット変換 ..        | A-12 |
| マルチバイト文字の取り扱い ..        | 2-13 |
| メッセージ・カタログの動的な変換 .....  | 3-21 |
| 文字分類 .....              | 2-16 |
| 文字列の比較 .....            | 2-17 |
| ユーザ・ロケールとの相違 ..         | 7-11 |
| ロケールでの使用 .....          | 2-5  |

## さ

|               |     |
|---------------|-----|
| サンプル・アプリケーション |     |
| 位置 .....      | 2-1 |
| サーバ           |     |

入力システムの起動..... 7-3

## し

シェアード・ライブラリ

methods ファイルでの指定... 6-63

ロケールのメソッド..... 6-62

ロケール・メソッドのサポー

ト ..... 6-29

シェル・スクリプト ..... 3-32

時刻

strftime 関数による変換 ..... 2-22

時刻値

書式付け ..... 2-21, A-5

時刻の書式

ロケール・ソース・ファイルでの定

義 ..... 6-26

実行時環境

ロケールのバインド..... 2-28

シフト状態 ..... 2-15

出力コンテキスト ..... 5-18

出力サーバ ..... 5-18

出力テキスト

書式付け ..... 2-26

照合

アジア系言語のサポート ..... 7-6

アルゴリズム ..... 2-18

使用する関数 ..... 2-18

性能の問題..... 2-18

レベル ..... 6-16

照合関数

性能の選択..... 2-18

照合順序

英語以外の文字の順序 ..... 1-5

ソースからの指定 ..... 6-15

ロケール・ソース・ファイルでの定

義 ..... 6-14

照合シーケンス

ロケール..... 1-6

照合テーブル

UDC 用の作成 ..... B-20

定数

英語以外の文字の使用 ..... 2-12

小数点..... 6-22

( 小数点文字 も参照 )

langinfo データベースからの取り出

し ..... 2-23

localeconv 関数による調査 ... 2-22

バリエーション ..... 2-19

小数点文字

金額値での定義 ..... 6-22

数値での定義 ..... 6-25

書式

金額値 ..... 2-22

書式指定子

出力テキスト文字列..... 2-26

入力テキスト文字列..... 2-27

書式付け

出力テキスト ..... 2-26

書式指定子 ..... 2-26

数値 ..... 2-22

入力テキスト ..... 2-27

日付と時刻..... 2-20, 2-21

メッセージ..... 2-26

シンボリック識別子

メッセージ・セット内の番号の置き

換え ..... 3-24

シンボリック名

gencat 入力用の変換 ..... 3-10

照合シンボルによる定義 ..... 6-18

文字マップ・ファイル ..... 6-4

## す

### 数値

カスタマイズされた書式付け 2-22

数値変換 ..... A-7

### スクリプト

メッセージ・カタログの使用 3-32

ロケール・データの取り出し 3-32

スタイルマネージャのバックドロップ・ファイル

CDE ..... 5-2

スタイルマネージャのパレット・ファイル  
CDE ..... 5-2

## せ

### 性能のトレードオフ

照合 ..... 2-18

### 正符号

localeconv 関数による調査 ... 2-22

金額値での定義 ..... 6-22

### 千単位の区切り文字

localeconv 関数による調査 ... 2-22

金額値の定義 ..... 6-22

数値での定義 ..... 6-25

バリエーション ..... 2-19

## そ

### ソフトウェア

国際化 ..... 1-1, 1-3

ソース・ファイル

メッセージ・カタログ ..... 3-2

### ソート

国際化規則..... 1-5

ハイフン付きの単語..... 6-16

### ソート規則

ロケール・ソース・ファイルでの定義 ..... 6-14

### ソート・ディレクティブ

2つのキーワード ..... 6-16

数..... 6-16

キーワード..... 6-15

## た

### 代替文字拡張

UTF-16..... 1-8

ダウンライン・ローダブル文字.. C-7

### タブ文字

メッセージ文字列..... 3-6

### 端末エミュレーション

プログラムのエスケープ・シーケンス ..... C-1

### 端末ドライバ

ユーザ定義文字の認識 ..... B-20

## ち

### 地域

文化的データ ..... 1-4

地域化..... 1-2

## つ

### 通貨記号

localeconv 関数による調査 ... 2-22

|                  |      |
|------------------|------|
| 国際通貨の定義 .....    | 6-22 |
| 国内通貨の定義 .....    | 6-22 |
| バリエーション .....    | 2-19 |
| ユーロ .....        | 2-19 |
| 月の名前             |      |
| ロケール・ソース・ファイルでの定 |      |
| 義 .....          | 6-26 |

## て

|                       |      |
|-----------------------|------|
| テキスト                  |      |
| 表示する curses ルーチン .... | 4-16 |
| 変換する curses ルーチン .... | 4-15 |
| テキスト入力                |      |
| X アプリケーションでの処理.       | 5-23 |
| テキストの表示               |      |
| X アプリケーションのフォントセッ     |      |
| ト .....               | 5-15 |
| テキスト表示                |      |
| 右から左へ.....            | 5-8  |
| テキスト文字列               |      |
| 長さと言語の統計 .....        | 3-14 |
| 翻訳のガイドライン.....        | 3-14 |
| テクニカル・リファレンス          |      |
| アジア系言語の文字の表示....      | 7-1  |
| データ                   |      |
| 国際化 .....             | 2-1  |
| プログラム・コードとの分離 .       | 2-1  |
| データタイプ・ファイル           |      |
| CDE .....             | 5-2  |
| データ・ファイル              |      |
| コードセット間での変換 ....      | 7-11 |
| データベース                |      |
| 統計情報の収集 .....         | B-21 |

## な

|               |      |
|---------------|------|
| 内部処理コード ..... | 2-14 |
|---------------|------|

## に

|                         |      |
|-------------------------|------|
| 入力サーバ                   |      |
| X アプリケーションでの決定. 5-24    |      |
| X アプリケーションの接続と接続の       |      |
| 解除 .....                | 5-24 |
| X アプリケーション呼び出し. 5-23    |      |
| アプリケーションとの実行....        | 7-3  |
| イベント・フィルタリング...         | 5-34 |
| 失敗の処理.....              | 5-37 |
| 省略時の .....              | 5-10 |
| 入力スタイル .....            | 5-25 |
| 入力スタイルをサポートするロケー        |      |
| ル .....                 | 5-27 |
| 前編集スタイル .....           | 5-27 |
| 入力システム                  |      |
| KeyPress .....          | 5-37 |
| KeyRelease .....        | 5-37 |
| イベントのフィルタリング... 5-37    |      |
| FocusIn と FocusOut..... | 5-37 |
| 入力スタイル                  |      |
| On-the-Spot .....       | 5-31 |
| 前編集スタイル .....           | 7-2  |
| 前編集スタイルの選択 .....        | 7-3  |

## ぬ

|            |      |
|------------|------|
| ヌル文字 ..... | 2-15 |
| 制限事項.....  | 2-15 |

## は

- バイト・オーダ
  - システムの省略時の設定 ..... 1-9
- パターン・ファイル ..... 3-19
- バックスラッシュ文字
  - メッセージ文字列 ..... 3-6

## ひ

- ビッグ・エンディアン
  - UTF-16 ..... 1-9
- 日付
  - strftime 関数による変換 ..... 2-22
  - 書式付け ..... 2-27, A-5
  - フォーマットの違い ..... 2-19
  - 文字列の生成 ..... 2-21
- 日付の書式
  - ロケール・ソース・ファイルでの元号の定義 ..... 6-28
  - ロケール・ソース・ファイルでの定義 ..... 6-26
- ビット・パターン
  - メッセージ文字列 ..... 3-6
- 否定応答
  - ロケールでの定義 ..... 6-19
- 表意文字
  - ソート ..... 7-6
  - 定義 ..... 7-4
  - リファレンス・ページ ..... 7-8
- 表示幅
  - マルチバイト幅の文字 ..... 6-60
- 標準 ..... 1-1
  - ( X/Open 標準 も参照 )

## 表フォーマット

- .TS マクロと .TE マクロ ..... 7-10

## ふ

- ファイル・コード ..... 2-13
- フォント
  - Motif ウィジェットによる検索 5-6
  - Motif 用の作成 ..... B-11
  - UDC のファイル ..... B-11
  - UDC ファイルの作成 ..... B-20
  - UDC ファイルの省略時の位置の設定 ..... 7-4
  - X アプリケーション用のコンパイル ..... B-24
  - システム・ソフトウェア用の作成 ..... B-11
  - ビットマップ
    - TrueType ..... 7-16
    - ユーザ定義グリフの作成 ..... B-9
  - フォント・エンコーディング
    - システムによる相違 ..... 5-19
    - 変換メカニズムを有効にする 5-20
  - フォント・グリフ
    - 置き換え ..... B-19
    - 実寸大での表示 ..... B-19
    - 照合値の指定 ..... B-20
    - 名前の指定 ..... B-20
    - 入力キー・シーケンスの指定 B-20
    - 描画 ..... B-18
    - 複数サイズの作成 ..... B-19
    - 複数プロトタイプの作成 .... B-19
    - 編集 ..... B-18
  - フォントセット ..... 5-12

Xt アプリケーションでのエンコー  
 ディング変換 ..... 5-4  
 Xt ライブラリ ..... 5-4  
 X アプリケーション ..... 5-12  
 エンコーディングの変換 ..... 5-19  
 作成と使用 ..... 5-13  
 テキストの表示 ..... 5-15  
 メトリックスの取得 ..... 5-15  
 フォントセットのエンコーディング  
 GL と GR の変換 ..... 5-19  
 フォント・ファイル  
 プリロード ..... B-23  
 フォント編集  
 cedit 画面 ..... B-11  
 終了 ..... B-20  
 フォント名  
 総称の使用の利点 ..... 5-14  
 フォント・レンダラ  
 TrueType フォント ..... 7-16  
 UDC ..... 7-15  
 アジア系の PostScript ..... 7-14  
 アジア系の PostScript 構成ファイ  
 ル ..... 7-15  
 符号拡張 ..... 6-36  
 部首 ..... 1-5  
 復帰文字  
 メッセージ文字列 ..... 3-6  
 負符号  
 localeconv 関数による調査 ... 2-22  
 金額値での定義 ..... 6-22  
 プログラミング手法  
 例 ..... 5-9  
 プログラム開発  
 国際化 ..... 1-1, 7-1  
 モジュール ..... 1-3

プログラム・コード  
 メッセージとの分離 ..... 1-3  
 文化的データ ..... 1-4  
 langinfo データベース ..... 2-20  
 小数点 ..... 2-19  
 千単位の区切り文字 ..... 2-19  
 通貨記号 ..... 2-19  
 データベース ..... 2-19  
 データベースからの抽出 ..... 2-20  
 日付フォーマット ..... 2-19

## へ

ヘルプ・ファイル  
 CDE ..... 5-2

## ほ

### 翻訳

trans ユーティリティ ..... 3-13  
 位置フォーマット ..... 3-17  
 語順 ..... 3-16  
 順序の指定 ..... 3-27  
 省略形 ..... 3-17  
 ソース・コメントの使用 ..... 3-13  
 ダイアログ・ボックスの設計 3-14  
 テキストとグラフィック ..... 3-14  
 テキスト文字列のガイドライ  
 ン ..... 3-14  
 文法規則 ..... 3-14  
 メッセージ・ガイドライン... 3-15  
 メッセージ・カタログ ..... 3-14  
 メッセージの要件 ..... 3-14  
 用語識別子 ..... 3-16  
 ポータブル文字セット

( PCS を参照 )

## ま

|                       |      |
|-----------------------|------|
| 前編集スタイル .....         | 7-2  |
| Off-the-Spot .....    | 7-2  |
| On-the-Spot .....     | 7-2  |
| Over-the-Spot .....   | 7-2  |
| Root Window .....     | 7-3  |
| 指定 .....              | 7-3  |
| 優先順位の設定 .....         | 7-3  |
| 前編集文字列                |      |
| X アプリケーションでの処理 .....  | 5-31 |
| 属性 .....              | 5-30 |
| マルチスレッド・アプリケーション      |      |
| errno の設定 .....       | 6-43 |
| マルチバイト・コードセット         |      |
| メソッドを使用する必要性 .....    | 6-29 |
| マルチバイト・データ            |      |
| UTF-8 ロケール .....      | 2-3  |
| マルチバイト文字 .....        | 1-6  |
| charmap .....         | 6-5  |
| 操作インタフェース .....       | 2-13 |
| テスト .....             | 2-12 |
| ワイド文字形式への変換 .....     | 2-13 |
| ワイド文字との比較 .....       | 2-13 |
| ワイド文字フォーマットへの変換 ..... | 6-29 |

## む

|              |      |
|--------------|------|
| 無視ファイル ..... | 3-19 |
|--------------|------|

## め

|                           |      |
|---------------------------|------|
| メソッド .....                | 6-29 |
| localedef コマンドへの指定 ..     | 6-65 |
| localedef の指定 .....       | 6-63 |
| mblen .....               | 6-40 |
| __mbstopcs .....          | 6-30 |
| mbstowcs .....            | 6-44 |
| __mbtopc .....            | 6-33 |
| mbtowc .....              | 6-46 |
| __pcstombs .....          | 6-38 |
| __pctomb .....            | 6-40 |
| wcstombs .....            | 6-50 |
| wcswidth .....            | 6-56 |
| wctomb .....              | 6-53 |
| wcwidth .....             | 6-59 |
| オプション .....               | 6-61 |
| オプションの一覧 .....            | 6-61 |
| オプションの作成 .....            | 6-61 |
| シェアード・ライブラリの作成 .....      | 6-62 |
| 省略時のアプリケーション ..           | 6-61 |
| 必須 .....                  | 6-30 |
| マルチバイト・コードセットでの要件 .....   | 6-29 |
| 利用可能性 .....               | 6-29 |
| メソッド・ファイル .....           | 6-1  |
| メッセージ .....               | 3-7  |
| アプリケーション・モジュールによる共有 ..... | 3-7  |
| 空文字列への変更 .....            | 3-11 |
| クォート区切り文字 .....           | 3-12 |
| 言語の制約 .....               | 2-24 |

|                                |          |                                |                  |
|--------------------------------|----------|--------------------------------|------------------|
| 最大長 .....                      | 3-11     | コメント行.....                     | 3-13             |
| 削除 .....                       | 3-11     | 受動態の動詞の作成.....                 | 2-26             |
| 識別子 .....                      | 3-10     | 使用する既存のプログラムの変換 .....          | 3-18             |
| シンボリック識別子.....                 | 3-24     | 使用するロケール.....                  | 2-25             |
| スタイルのガイドライン .....              | 3-14     | スクリプトのアクセス .....               | 3-32             |
| セット内の順序 .....                  | 3-10     | 性能上の問題 .....                   | 3-30             |
| 前後のスペース .....                  | 3-6      | 設計と保守に関する留意点...                | 3-27             |
| 特殊文字のコーディング .....              | 3-6      | ソース形式への変換.....                 | 3-32             |
| プログラム・コードとの分離 .                | 1-3      | ソース・ファイル.....                  | 3-13             |
| プログラムへの読み込み .....              | 3-40     | ソース・ファイルからの作成 .                | 3-5              |
| 保守 .....                       | 3-7      | ソース・ファイル内の文字列を<br>クォートする ..... | 3-6              |
| 保守の設計方針 .....                  | 3-28     | ソース・ファイルの編集 .....              | 3-20             |
| メッセージ・カタログからの表<br>示 .....      | 3-33     | 動的なコードセット変換 .....              | 3-21             |
| 文字列の作成 .....                   | 2-26     | 内容の表示.....                     | 3-32             |
| 要素の順序.....                     | 2-26     | 日付の書式.....                     | 2-27             |
| メッセージ・エントリ<br>形式 .....         | 3-10     | 標準位置以外へのインストー<br>ル .....       | 3-35             |
| メッセージ・カタログ.....                | 1-4, 3-7 | ファイル・オープン失敗の検<br>出 .....       | 3-39             |
| CDE .....                      | 5-2      | ファイル名拡張子.....                  | 3-26             |
| gencat コマンド.....               | 3-26     | 複数と 1 つのソースの組み合わ<br>せ .....    | 3-31             |
| langinfo データベースとの相違<br>点 ..... | 3-1      | プログラム・ソースのコンパイ<br>ル .....      | 3-28             |
| Motif アプリケーション.....            | 1-4      | プログラムのアクセス .....               | 3-34             |
| NLSPATH 環境変数.....              | 3-34     | プログラム変換用のフローチャー<br>ト .....     | 3-19             |
| root のアカウント下 .....             | 3-39     | プログラム・モジュールごと                  | 3-28             |
| set ディレクティブ.....               | 3-8      | 翻訳 .....                       | 3-13, 3-20, 3-21 |
| アプリケーションごと .....               | 3-29     | 翻訳者に役立つコメント .....              | 3-13             |
| 位置 .....                       | 2-25     | ポータビリティ .....                  | 3-26             |
| 一般規則 .....                     | 3-5      | メッセージ・セットの削除....               | 3-9              |
| 英語以外の定数の定義 .....               | 2-12     |                                |                  |
| 空白行 .....                      | 3-7      |                                |                  |
| クローズ.....                      | 3-40     |                                |                  |
| 語順の変更.....                     | 2-26     |                                |                  |
| 異なるロケールのための生成                  | 3-22     |                                |                  |



|                  |            |
|------------------|------------|
| メッセージ・セットの順序     | 3-8        |
| メッセージの削除         | 3-11       |
| メッセージの取り出し       | 2-24       |
| ロケールとの比較         | 3-1        |
| メッセージ識別子         |            |
| シンボルの利点          | 3-27       |
| メッセージ・システム       | 2-1        |
| X/Open 標準        | 2-24       |
| メッセージ・セット        | 3-7        |
| (メッセージ・カタログも参照)  |            |
| 削除               | 3-9, 3-11  |
| 識別子の規則           | 3-8        |
| 識別子の指定           | 3-8        |
| 省略時の             | 3-8        |
| シンボリック識別子        | 3-24       |
| すべてのメッセージの置き換え   |            |
| え                | 3-12       |
| 利点               | 3-7        |
| メッセージ・ソース・ファイル   |            |
| mkcatdefs による前処理 | 3-22, 3-23 |
| 行の継続             | 3-7        |
| シンボリック名          | 3-10       |
| 内容               | 3-5        |
| フィールドの区切り        | 3-6        |
| プログラムごと          | 3-29       |
| プログラム・モジュールごと    | 3-28       |
| メッセージの順序付け       | 3-5        |
| メッセージの置き換え       |            |
| gencat コマンド      | 3-12       |
| mkcatdefs コマンド   | 3-12       |
| メッセージの削除         |            |
| set ディレクティブの指定   | 3-11       |

|                   |      |
|-------------------|------|
| 識別子と他の文字          | 3-11 |
| シンボリック識別子         | 3-11 |
| 数値識別子             | 3-11 |
| メッセージ・ファイル        |      |
| 16 進値の使用          | 3-6  |
| 8 進値の使用           | 3-6  |
| 位置フォーマット          | 3-17 |
| 改行文字              | 3-7  |
| バックスラッシュ          | 3-6  |
| 複数の quote ディレクティブ | 3-13 |
| 保守のガイドライン         | 3-31 |
| 利点                | 2-24 |
| メッセージ文字列          |      |
| 区切り文字の指定          | 3-12 |
| ソース・ファイルへの抽出      | 3-18 |

## も

|                |      |
|----------------|------|
| 文字             |      |
| 大文字と小文字の変換     | 2-17 |
| クラスの識別         | 2-16 |
| 照合             | 2-18 |
| 状態依存のエンコーディング  | 2-15 |
| マルチバイト         | 1-6  |
| 変換用メソッドの作成     | 6-29 |
| 文字データ型         | 1-6  |
| ロケールのエンコーディング  | 6-4  |
| ワイド            | 1-6  |
| 文字クラス          |      |
| Unicode 定義のテスト | A-3  |
| XSH 定義のテスト     | A-3  |
| ロケールでの定義       | 6-9  |
| 文字コード          |      |
| 特徴             | 1-3  |

|                       |      |
|-----------------------|------|
| 変換 .....              | 1-4  |
| 文字サイズ                 |      |
| 16 ビット .....          | 1-8  |
| 32 ビット .....          | 1-8  |
| 文字照合関数 .....          | A-5  |
| 文字セット .....           | 1-5  |
| (コードセット も参照)          |      |
| UCS .....             | 1-8  |
| ポータブル .....           | 1-7  |
| 文字属性データベース .....      | 7-4  |
| (UDC データベース も参照)      |      |
| 文字のソート                |      |
| 異なる言語 .....           | 7-6  |
| 文字のプロパティ              |      |
| ロケールでの定義 .....        | 6-9  |
| 文字の列 .....            | 1-6  |
| 文字分類                  |      |
| 関数 .....              | 2-16 |
| 文字マップ .....           | 6-1  |
| (charmap ファイル も参照)    |      |
| サンプル・ソース・ファイル . D-1   |      |
| マルチバイト文字 .....        | 6-5  |
| マルチバイト文字ファイルの要件 ..... | 6-1  |
| 文字列 .....             | 1-6  |
| (文字の列 も参照)            |      |
| 空 .....               | 1-6  |
| 読み取る curses ルーチン .... | 4-12 |
| 文字列処理関数 .....         | A-10 |
| 文字列の比較 .....          | 2-17 |
| 文字列ファイル .....         | 3-18 |

## ゆ

ユニバーサル文字セット

|                      |      |
|----------------------|------|
| (UCS を参照)            |      |
| ユーザ定義文字              |      |
| (UDC を参照)            |      |
| ユーロのサポート             |      |
| コードセット .....         | 2-9  |
| ユーロ文字                |      |
| LC_MONETARY カテゴリ ... | 6-24 |

## よ

|                         |      |
|-------------------------|------|
| 曜日の名前                   |      |
| ロケール・ソース・ファイルでの定義 ..... | 6-26 |

## ら

|               |     |
|---------------|-----|
| ライブラリ関数 ..... | 2-1 |
|---------------|-----|

## り

|                   |      |
|-------------------|------|
| リソース・データベース       |      |
| 地域化された処理 .....    | 5-22 |
| リソース・ファイル         |      |
| CDE .....         | 5-2  |
| リテラル              |      |
| PCS 文字 .....      | 2-12 |
| リトル・エンディアン        |      |
| UTF-16 .....      | 1-9  |
| リファレンス・ページ        |      |
| CDE .....         | 5-2  |
| 表意文字 .....        | 7-8  |
| プリント .....        | 7-10 |
| 翻訳ファイルの位置 .....   | 7-10 |
| リファレンス・ページのフォーマット |      |
| 行頭禁則文字 .....      | 7-8  |

行末禁則文字 ..... 7-8

## れ

例

ximdemo アプリケーション ... 5-9

## ろ

ロケール

CDE でのサポートを有効にする

る ..... 5-1

dense コード ..... 2-2

dense コードと Unicode の間の切り

替え ..... 2-2

dense コードと Unicode の類似

点 ..... 2-4

Motif アプリケーションでの設

定 ..... 5-5

nroff コマンドのサポート ..... 7-7

setlocale 指定子 ..... 2-29

Unicode コード ..... 2-2

Unicode と dense コードの

charmap ..... 6-19

Unicode と dense コードのソー

ス ..... 6-19

UTF-32 処理コードの提供 ..... 2-9

UTF-8 ..... 2-3

Xt アプリケーションでの設定 ..... 5-3

X アプリケーション ..... 5-9

X アプリケーションでの設定 ..... 5-11

X アプリケーションのオブジェク

ト ..... 5-10

X アプリケーションのフォントセッ

ト ..... 5-12

位置 ..... 6-65

カテゴリ ..... 2-28

カテゴリの定義 ..... 6-7

間接的なシステム・サポート ..... 5-2

国際化ソフトウェア構成ユーティリ

ティによる設定 ..... 2-2

コードポイント・マッピング ..... 2-4

サイズの縮小 ..... 6-17

サンプル・ソース・ファイル ..... D-1

実行時の初期化 ..... 2-28

照合シーケンス ..... 1-6

情報の表示 ..... 3-33

省略時のシステムの位置 ..... 6-65

スクリプトからのデータの取り出

し ..... 3-32

ソース・ファイル

charmap ファイル ..... 6-2

ロケール定義ファイル ..... 6-7

直接的なシステム・サポート ..... 5-2

地域化システムでの提供 ..... 2-5

重複する定義のチェック ..... 6-65

直接的なシステム・サポートを有効

にする ..... 5-3

テスト ..... 6-65

特定のカテゴリの設定の変更 ..... 2-31

名前拡張子 ..... 6-65

名前の拡張 ..... 2-29

名前の構成要素 ..... 2-29

標準以外の文字 ..... 6-13

標準のシステムによる提供 ..... 2-5

プログラム内での変更 ..... 2-30

- プログラムのバインド ..... 2-29
- メソッドが必要な場合 ..... 6-29
- メッセージ・カタログとの比較 3-1
- 文字分類 ..... 2-16
- ロケール・カテゴリ
  - LC\_COLLATE ..... 6-14
  - LC\_CTYPE ..... 6-9
  - LC\_MESSAGES ..... 6-19
  - LC\_MONETARY ..... 6-21
  - LC\_NUMERIC ..... 6-24
  - LC\_TIME ..... 6-25
  - 省略した場合の定義 ..... 6-9
- ロケール・ソース・ファイル .... 6-1
  - エスケープ文字 ..... 6-8
  - コメントの指定 ..... 6-8
- ロケール定義
  - サンプル・ソース・ファイル . D-7
- ロケール・バリエーション
  - 設定 ..... 2-30
- ロケール・ファイル
  - 名前の重複の解決 ..... 6-66
  - プログラムに認識させる ..... 2-30
- ロケール変数
  - 設定 ..... 1-1
- ロケール名 ..... 1-1

サフィックスによる割り当て . 7-6

## わ

- ワイド文字 ..... 1-6
  - キーボードから読み取る curses ルーチン ..... 4-14
  - 削除する curses ルーチン .... 4-8
  - 省略時のサイズ ..... 2-14
  - 挿入する curses ルーチン .... 4-3
  - 追加する curses ルーチン .... 4-2
  - マルチバイト文字との比較... 2-13
  - 読み取る curses ルーチン .... 4-9
- ワイド文字・エンコーディング
  - Unicode ロケール ..... 2-3
- ワイド文字データ型
  - WPI サポート ..... A-1
- ワイド文字のエンコーディング
  - ctype の使用 ..... 2-17
  - dense コード・ロケール ..... 2-3
- ワイド文字列 ..... 1-6
  - 挿入する curses ルーチン .... 4-7
  - 追加する curses ルーチン 4-4, 4-5
  - 読み取る curses ルーチン ... 4-10, 4-11

## Tru64 UNIX ドキュメントの購入方法

Tru64 UNIX ドキュメントのご購入については、弊社担当営業または日本ヒューレット・パッカートの各営業所/代理店にお問い合わせください。

各ドキュメント・キットの注文番号は以下のとおりです。ドキュメント・キットに含まれるマニュアルの内容については『ドキュメント概要』を参照してください。

| キット名                                              | 注文番号        |
|---------------------------------------------------|-------------|
| Tru64 UNIX Documentation CD-ROM                   | QA-6ADAA-G8 |
| Tru64 UNIX Documentation Kit                      | QA-6ADAA-GZ |
| End User Documentation Kit                        | QA-6ADAB-GZ |
| - Startup Documentation Kit                       | QA-6ADAC-GZ |
| - General User Documentation Kit                  | QA-6ADAD-GZ |
| - System and Network Management Documentation Kit | QA-6ADAE-GZ |
| Developer's Documentation Kit                     | QA-6ADAF-GZ |
| Reference Pages Documentation Kit                 | QA-6ADAG-GZ |
| TruCluster Server Documentation Kit               | QA-6BRAA-GZ |
| Tru64 UNIX 日本語ドキュメント・キット                          | QA-6ADJB-GZ |
| スタートアップ・ドキュメント・キット                                | QA-6ADJC-GZ |
| 一般ユーザ・ドキュメント・キット                                  | QA-6ADJD-GZ |
| システム/ネットワーク管理ドキュメント・キット                           | QA-6ADJE-GZ |
| プログラミング・ドキュメント・キット                                | QA-6ADJF-GZ |
| CDE 翻訳ドキュメント・キット                                  | QA-6ADJG-GZ |
| TruCluster Server 日本語ドキュメント・キット                   | QA-05SJA-GZ |
| Advanced Server for UNIX 日本語ドキュメント・キット            | QA-5U2JA-GZ |



# マニュアルに対するご意見

## Tru64 UNIX

国際化ソフトウェア・プログラミング・ガイド

AA-RK3SC-TE

弊社のマニュアルに関して、ご意見、ご要望、または内容の不明確な部分など、お気づきの点がございましたら、下記にご記入の上、弊社社員にお渡しくださるようお願い申し上げます。

マニュアルの採点：

|                   | 大変良い                     | 良い                       | 普通                       | 良くない                     |
|-------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 正確さ(説明どおりに動作するか)  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 情報量(十分か)          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 分かり易さ             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| マニュアルの構成          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 図(役立つか)           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 例(役立つか)           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 索引(項目の検索性)        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ページ・レイアウト(情報の検索性) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

内容の不明確な部分がありましたら、以下にご記入ください：

ペー ジ

|  |  |
|--|--|
|  |  |
|  |  |
|  |  |
|  |  |

その他お気づきの点がございましたら、以下にご記入ください：

|  |
|--|
|  |
|  |
|  |
|  |

ご使用のソフトウェアのバージョン： \_\_\_\_\_

貴社名/部課名 \_\_\_\_\_

御名前 \_\_\_\_\_

記入日 \_\_\_\_\_

(注) 当用紙を受け取った弊社社員は、すみやかに下記にお送りください。

ビジネスクリティカルシステム統括本部 **BCS** 技術本部 **Alpha** ソフトウェア技術部