

Tru64 UNIX

---

System V MNLS から  
Tru64 UNIX への  
国際化機能移行ガイド

AA-RK3WA-TE

1999 年 10 月

オペレーティング・システム: Tru64 UNIX V5.0

コンパックコンピュータ株式会社

---

1999 年 10 月

本書の著作権はコンパックコンピュータ株式会社が保有しており、本書中の解説および図、表はコンパック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きください。万一、本書の記述に誤りがあった場合でも、コンパックは一切その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。

コンパックは、コンパックまたはコンパック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。

© Compaq Computer Corporation 1999.

All Rights Reserved.  
Printed in Singapore.

Motif, OSF および OSF/1 は米国 Open Software Foundation 社の商標です。

POSIX は IEEE の商標です。

UNIX は X/Open カンパニーリミテッドが独占的にライセンスしている米国ならびに他の国における登録商標です。

---

# 目次

まえがき .....	vii
1 System V MNLS からの国際化機能の移行方法	
1.1 MNLS マイグレーション・キットを使用する方法 .....	1-1
1.1.1 特徴 .....	1-1
1.1.2 相違点の解決方法 .....	1-2
1.2 XPG4 国際化インタフェースを使用する方法 .....	1-2
1.2.1 特徴 .....	1-3
1.2.2 相違点の解決方法 .....	1-3
2 MNLS マイグレーション・キットを使用した国際化機能の移行	
2.1 MNLS マイグレーション・キット .....	2-1
2.1.1 MNLS マイグレーション・キットの構成 .....	2-2
2.1.2 MNLS マイグレーション・キットで提供される機能 .....	2-3
2.1.2.1 Tru64 UNIX 上に存在しない System V MNLS 独自の関数 .....	2-5
2.1.2.2 System V MNLS と Tru64 UNIX で名称が異なる関数を Tru64 UNIX の関数に置き換えるマクロ .....	2-6
2.1.2.3 EUC コードセットの画面制御情報を取得するマクロ .....	2-6
2.1.2.4 gettxt 関数 .....	2-6
2.2 移行の手順 .....	2-6
2.3 プログラム修正規模の把握 .....	2-7
2.4 プログラムの修正 .....	2-8
2.4.1 MNLS マイグレーション・キットで未対応の関数に関する修正 .....	2-8
2.4.2 書式付き入出力関数におけるワイド文字制御の変換仕様の修正 .....	2-12
2.4.3 EUC コードセットに依存した処理に関する修正 .....	2-12
2.4.4 curses ライブラリ関数に関する修正 .....	2-13

2.5	プログラムのコンパイルおよびリンク .....	2-13
2.5.1	コンパイル・オプションに MNLS マイグレーション・キット のヘッダ・ファイルの格納先を指定する .....	2-13
2.5.2	リンク・オプションに MNLS マイグレーション・キットのラ イブラリの格納先を指定する .....	2-14
2.6	メッセージ・ファイルの変換 .....	2-14
2.7	メッセージ・カタログの生成 .....	2-15
2.8	生成されたメッセージ・カタログの内容の確認 .....	2-15
3	XPG4 国際化インタフェースを使用した国際化機能の移行	
3.1	国際化機能移行支援ツール .....	3-1
3.2	移行の手順 .....	3-1
3.3	プログラム修正規模の把握 .....	3-2
3.4	プログラムの修正 .....	3-2
3.5	プログラムのコンパイル .....	3-3
3.6	メッセージ・ファイルの変換 .....	3-3
3.7	メッセージ・カタログの生成 .....	3-3
3.8	生成されたメッセージ・カタログの内容の確認 .....	3-4
A	MNLS から XPG4 への国際化機能移行ガイド	
A.1	プログラムの修正方法 .....	A-1
A.1.1	ヘッダ・ファイル .....	A-1
A.1.2	ライブラリ関数 .....	A-2
A.1.2.1	System V MNLS 固有の関数の対応について .....	A-6
A.1.2.2	書式付き入出力関数におけるワイド文字制御の変換 仕様 .....	A-9
A.1.3	EUC コードセットに依存した処理 .....	A-10
A.1.4	curses ライブラリ関数 .....	A-10
A.2	メッセージ・ファイルの移行 .....	A-11
A.3	プログラム実行時の設定 .....	A-11

## B プログラムの移行例

B.1	サンプル・プログラム .....	B-1
B.2	MNLS マイグレーション・キットを利用した移行例 .....	B-3
B.2.1	プログラム移行支援ツールによる修正箇所の調査 .....	B-3
B.2.2	プログラム移行支援ツールで検出できない修正箇所の調査 .....	B-4
B.2.3	プログラムの修正 .....	B-4
B.2.4	メッセージ・ファイルの変換 .....	B-7
B.2.5	プログラムのコンパイル .....	B-7
B.2.6	プログラムの実行 .....	B-7
B.3	XPG4 国際化インタフェースへの書き換えによる移行例 .....	B-8
B.3.1	プログラム移行支援ツールによる修正箇所の調査 .....	B-8
B.3.2	プログラム移行支援ツールで検出できない修正箇所の調査 .....	B-9
B.3.3	プログラムの修正 .....	B-10
B.3.4	メッセージ・ファイルの変換 .....	B-17
B.3.5	プログラムのコンパイル .....	B-17
B.3.6	プログラムの実行 .....	B-18

## C リファレンス・ページ

getwidth .....	C-2
CODESETNO .....	C-4
msgtrn .....	C-5
chkmnls .....	C-7

## 図

2-1	マイグレーション・キットの構造 .....	2-2
2-2	MNLS マイグレーション・キットで提供される機能 .....	2-3

## 表

2-1	マイグレーション・キットでカバーする機能の一覧 .....	2-4
2-2	Tru64 UNIX のライブラリでカバーする機能の一覧 .....	2-4
A-1	ヘッダ・ファイル比較一覧 .....	A-1
A-2	ライブラリ関数比較一覧 .....	A-3
B-1	ディレクトリ構成 .....	B-1

B-2	分類.....	B-2
-----	---------	-----

---

## まえがき

### 本書の目的

本書は、System V MNLS(Multi-National Language Supplement) の国際化機能と Tru64 UNIX の国際化機能を比較し、System V MNLS の国際化機能を使用したアプリケーション (以下 MNLS アプリケーションと略します) を Tru64 UNIX 上に移行する際に必要な情報を提供します。

### 本書の対象読者

本書は、以下の読者を対象としています。

- MNLS アプリケーションを Tru64 UNIX に移行することを検討しているソフトウェア開発者
- System V MNLS の国際化機能に精通していて、Tru64 UNIX あるいは XPG4 の国際化機能の知識を必要としているソフトウェア開発者および一般ユーザ

### 本書の構成

本書の構成は次のとおりです。

第 1 章	System V MNLS からの国際化機能の移行方法
第 2 章	MNLS マイグレーション・キットを使用した国際化機能の移行
第 3 章	XPG4 国際化インタフェースを使用した国際化機能の移行
付録 A	MNLS から XPG4 への国際化機能移行ガイド
付録 B	プログラムの移行例
付録 C	リファレンス・ページ

## 関連資料

以下の Tru64 UNIX のマニュアルには、本書の内容と関連する情報が含まれています。

- 『日本語機能ガイドブック』
- 『プログラミング・ガイド』
- 『国際化ソフトウェア・プログラミング・ガイド』

System V MNLS および XPG4 の詳細については、以下の本を参照してください。

- 『UNIX System V リリース 4 国際化機能 (MNLS) 機能説明書[英語版]』 (共立出版)
- 『X/Open Portability Guide Issue 4』 (X/Open Company Ltd.)

## 規約

本書では次の表記法を使用します。

Ctrl/x	Ctrl キーを押しながら別のキーを押すことを示します。
PFn	数字キーパッドの PFn と書かれたキーを押すことを示します。n は 1, 2, 3, または 4 です。
PF1 x	まず PF1 と書かれたキーを押して離し、次に別のキーを押して離すことを示します。
<span style="border: 1px solid black; padding: 2px;">Return</span>	四角で囲まれたキー名は、キーボードのキーを押すことを示します (本文中ではキー名は四角で囲まれていません)。
.	縦の省略記号は、コード例またはコマンド・フォーマットから項目が省略されていることを示します。省略されているのは、説明されている主題にとってその項目が重要でないためです。
...	フォーマットの説明で使用されている横の省略記号は、その前の項目が 1 回以上反復されることを示します。
[ ]	フォーマットの説明で、括弧に入っている要素は任意指定であることを示します。選択項目は何も選択しなくてもよく、または 1 つ、複数、あるいはすべてを選択してもかまいません。



	ユーティリティ・フォーマット行内にある 2 つのオプションの間の縦線は、コマンド行でオプションのうち一度にいずれか 1 つしか使用できないことを示します。この縦線の用法は、コマンド行内で 1 つのユーティリティから別のユーティリティに出力をパイプするリダイレクト演算子とは異なります。
太字	新しい用語、オプション、またはパラメータを示します。本書のオンライン版でユーザ入力を示すためにも使用されます。
数字	特に断りがない限り、本文中の数字はすべて 10 進数とします。非 10 進基数 (2 進数、8 進数、または 16 進数) の場合はそれを明示します。



---

## System V MNLS からの国際化機能の移行方法

MNLS アプリケーションの国際化機能の Tru64 UNIX への移行 (以下, 「アプリケーションの国際化機能の移行」と呼びます) は, 次の 2 つの方法があります。

- MNLS マイグレーション・キットを使用する方法
- XPG4 国際化インタフェースを使用する方法

この章では, この 2 つの方法の特徴と概要について説明します。

---

### 1.1 MNLS マイグレーション・キットを使用する方法

日本語 Tru64 UNIX で提供される MNLS マイグレーション・キットでは, Tru64 UNIX の国際化機能を使用して System V MNLS の機能を実現します。この MNLS マイグレーション・キットを使用すると, 容易にアプリケーションの国際化機能の移行ができます。

#### 1.1.1 特徴

MNLS マイグレーション・キットを使用したアプリケーションの国際化機能の移行では, 次のような特徴があります。

- MNLS マイグレーション・キットを使用することにより, 段階的な移行が可能になります。つまり, MNLS アプリケーションを, EUC コードセットの動作環境で移行を行い, その後, 他のコードセットへの対応が必要になった時点で, XPG4 の国際化機能へ移行することが可能となります。
- 最小の修正で, MNLS アプリケーションを Tru64 UNIX の国際化機能の環境へ移行できます。つまり, 主要な System V MNLS の関数は, マイグレーション・キットで対応しているので, プログラムの修正は必要ありません。

- MNLS マイグレーション・キットが対応しているロケールは ja\_JP.eucJP です。つまり、MNLS マイグレーション・キットが提供する機能は、EUC コードセットにのみ有効です。

### 1.1.2 相違点の解決方法

MNLS マイグレーション・キットは、System V MNLS と Tru64 UNIX の国際化機能のライブラリ関数の相違点を次のようにして解決します。

- 関数の名称が異なるもの  
System V MNLS には、Tru64 UNIX のライブラリ関数と関数の名称が違うだけのものがあります。これらの関数は、マクロにより Tru64 UNIX のライブラリ関数に置き換えます。
- Tru64 UNIX 上に存在しない System V MNLS 固有のワイド文字の分類関数  
マクロにより iswctype 関数に置き換えます。iswctype 関数の引数として、日本語 Tru64 UNIX で提供している文字クラスを使用します。
- 参照するヘッダ・ファイルの名称が異なるもの  
Tru64 UNIX 上に存在しない System V MNLS 固有のヘッダ・ファイルを追加しています。また、その追加したヘッダ・ファイルの中で、Tru64 UNIX のライブラリ関数で参照するヘッダ・ファイルを展開しています。
- 関数のプロトタイプ宣言において、引数またはリターン値の型の宣言が異なるもの  
型の違いは、Tru64 UNIX では特に問題になりません。

---

## 1.2 XPG4 国際化インタフェースを使用する方法

Tru64 UNIX は、XPG4 に適合した国際化インタフェースを実装しています。この XPG4 国際化インタフェースを使用してアプリケーションの国際化機能を移行すれば、Tru64 UNIX 上での動作だけでなく、XPG4 に適合した OS での動作が保証され、プログラムのポータビリティが広がります。

### 1.2.1 特徴

XPG4 国際化インタフェースを使用したアプリケーションの国際化機能の移行では、次のような特徴があります。

- XPG4 国際化機能を使用することにより、各コードセットに依存した処理 (たとえば、EUC コードセットやシフト JIS コードセットに固有の処理) を、アプリケーションから排除することが可能になります。しかし、これを行うには、アプリケーションを書き換える必要があります。特に、EUC コードセットに依存した記述は取り除く必要があります。たとえば、EUC コードセット固有のビットパターンを前提としたロジックなどです。
- 一つのインタフェースで、いろいろなロケールが使用可能となります。XPG4 国際化インタフェースは、EUC コードセット、シフト JIS コードセット 共に使用可能です。アプリケーションを修正せずに対応が可能です。
- XPG4 に適合した他の OS への移行ができます。XPG4 に適合したプログラミング環境を持つ OS であれば、国際化機能に関しては、プログラムの修正が不要になります。

### 1.2.2 相違点の解決方法

XPG4 国際化インタフェースを使用する方法での、System V MNLS と Tru64 UNIX の国際化機能のライブラリ関数の違いを解決する方法は、アプリケーションのプログラムから System V MNLS 固有の記述箇所を取り除き、XPG4 国際化インタフェースを使用した記述に置き換えることです。それは具体的には次のとおりです。

- 関数の名称が異なるもの  
System V MNLS には、Tru64 UNIX のライブラリ関数と関数の名称が違うだけのものがあります。これらの関数は、関数名を書き直します。
- Tru64 UNIX 上に存在しない System V MNLS 固有のワイド文字の分類関数  
iswctype 関数を使用して書き換えます。iswctype 関数の引数として、日本語 Tru64 UNIX で提供している文字クラスを指定します。ただし、この文字クラスは XPG4 で定義された文字クラスではありませんから、他の OS へ移行する

## System V MNLS からの国際化機能の移行方法

### 1.2 XPG4 国際化インタフェースを使用する方法

際には、`iswctype` 関数の引数として使用した文字クラスと同じものを移行先の OS で定義する必要があります。

- 参照するヘッダ・ファイルの名称が異なるもの

ヘッダ・ファイルの展開箇所を XPG4 国際化インタフェースの仕様に合わせて修正します。

- 関数のプロトタイプ宣言において、引数またはリターン値の型の宣言が異なるもの

型の違いは、ポータビリティを考慮した場合は問題になりますから、リターン値を格納する変数や、引数の値を格納する変数の型は、XPG4 国際化インタフェースの仕様に合わせて書き換える必要があります。

具体的な移行方法の説明については、第 2 章または第 3 章を参照してください。また、System V MNLS と Tru64 UNIX の国際化機能のライブラリ関数の違いについては、付録 A を参照してください。書き換えについての詳しい説明も、付録 A を参照してください。

---

## MNLS マイグレーション・キットを使用した国際化機能の移行

この章では、MNLS マイグレーション・キットの概要、MNLS マイグレーション・キットを使用した MNLS アプリケーションの国際化機能の移行方法について説明します。

---

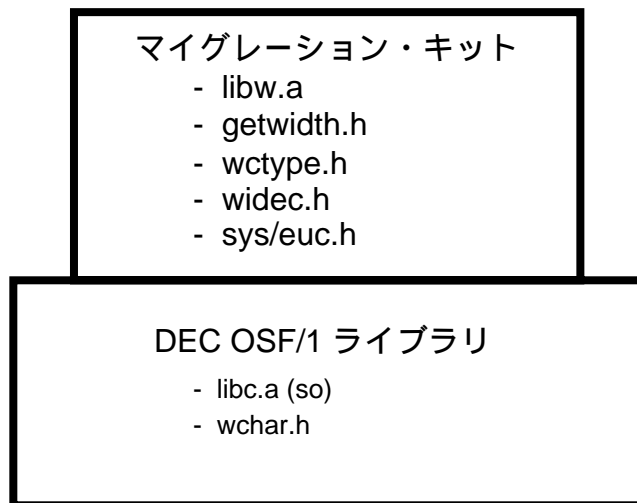
### 2.1 MNLS マイグレーション・キット

MNLS アプリケーションは、そのままでは Tru64 UNIX でコンパイルはできません。そこで、最低限の修正でコンパイルが可能なように、ヘッダ・ファイルと、ライブラリが追加されています。これを MNLS マイグレーション・キットと呼びます。図 2-1 に示すように、MNLS マイグレーション・キットは Tru64 UNIX のライブラリを使用して作成されています。

## MNLS マイグレーション・キットを使用した国際化機能の移行

### 2.1 MNLS マイグレーション・キット

図 2-1 マイグレーション・キットの構造



この節では、この MNLS マイグレーション・キットを使用して MNLS アプリケーションを Tru64 UNIX へ移行する手順について説明します。

#### 2.1.1 MNLS マイグレーション・キットの構成

MNLS マイグレーション・キットは次のファイルで構成されています。

- ヘッダ・ファイル

```
/usr/i18n/include/svr4/getwidth.h  
/usr/i18n/include/svr4/wctype.h  
/usr/i18n/include/svr4/widec.h  
/usr/i18n/include/svr4/sys/euc.h
```

- ライブラリ

```
/usr/i18n/lib/svr4/libw.a
```

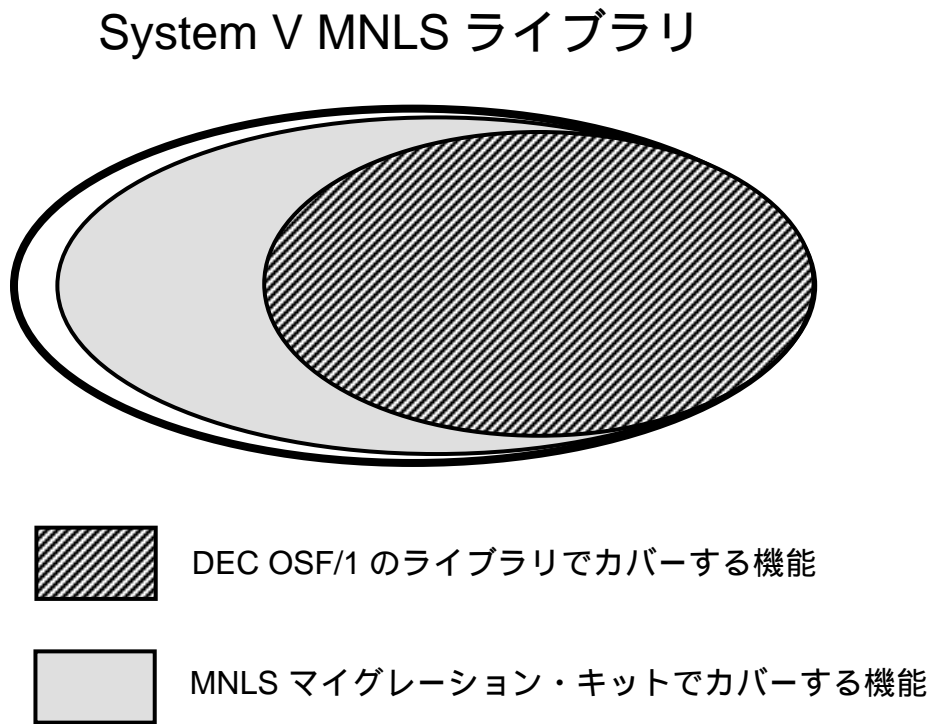
## 2-2 MNLS マイグレーション・キットを使用した国際化機能の移行



### 2.1.2 MNLS マイグレーション・キットで提供される機能

図 2-2 に示すように，MNLS マイグレーション・キットは，Tru64 UNIX のライブラリではカバーできない System V MNLS ライブラリの機能を提供します。

図 2-2 MNLS マイグレーション・キットで提供される機能



マイグレーション・キットでカバーする機能の一覧を表 2-1 に，Tru64 UNIX のライブラリでカバーする機能の一覧を表 2-2 に示します。

MNLS マイグレーション・キットを使用した国際化機能の移行  
2.1 MNLS マイグレーション・キット

表 2-1 マイグレーション・キットでカバーする機能の一覧

関数名，ヘッダ・ファイル名				
getwidth.h	sys/euc.h	wctype.h	widec.h	CODESETNO
getwidth	getws	isenglish	isideogram	isnumber
isphonogram	isspecial	iswascii	putws	strtows
wscat	wchr	wscmp	wscpy	wscspn
wslen	wsncat	wsncmp	wsncpy	wspbrk
wsrchr	wsspn	wstok	wstostr	

表 2-2 Tru64 UNIX のライブラリでカバーする機能の一覧

関数名，ヘッダ・ファイル名				
ctype.h	locale.h	stdarg.h	stdio.h	stdlib.h
fgetwc	fgetws	fprintf	fputwc	fputws
fscanf	getopt	gettext	getwc	getwchar
iswalnum	iswalpha	iswcntrl	iswdigit	iswgraph
iswlower	iswprint	iswpunct	iswspace	iswupper
iswxdigit	mblen	mbstowcs	mbtowc	printf
putwc	putwchar	scanf	setlocale	sprintf
sscanf	towlower	towupper	ungetwc	vfprintf
vprintf	vsprintf	wcstombs	wctomb	

MNLS マイグレーション・キットで提供される機能は次のとおりです。

- Tru64 UNIX 上に存在しない System V MNLS 独自の関数
- System V MNLS と Tru64 UNIX で名称が異なる関数を Tru64 UNIX の関数に置き換えるマクロ
- EUC コードセットの画面制御情報を取得するマクロ

#### 2.1.2.1 Tru64 UNIX 上に存在しない System V MNLS 独自の関数

MNLS マイグレーション・キットで提供される関数は次のものです。

- wstostr 関数

基本的に System V MNLS の関数と互換です。扱える文字列の長さに制限がありますが、通常の処理では問題となりません。wstostr 関数は、変換された複数バイト文字列のバイト数が int の最大値 (2147483647) まで処理を行うと、変換を中止します。この時、変換された複数バイト文字列の後ろにはヌル文字は付加されません。また、関数のリターン値は正常終了と同じで、変換された複数バイト文字列の先頭のポインタを返します。

- strtows 関数

基本的に System V MNLS の関数と互換です。扱える文字列の長さに制限がありますが、通常の処理では問題となりません。strtows 関数は、変換されたワイド文字列の文字数が int の最大値 (2147483647) まで処理を行うと、変換を中止します。この時、変換されたワイド文字列の後ろにはヌルのワイド文字は付加されません。また、関数のリターン値は正常終了と同じで、変換されたワイド文字列の先頭のポインタを返します。

- getws 関数

System V MNLS の関数と互換です。

- putws 関数

System V MNLS の関数と互換です。

- getwidth 関数

関数の形式は System V MNLS の関数と互換ですが、この関数は EUC コードセットに強く依存しており、Tru64 UNIX の国際化機能では完全な互換性は実現できません。使用上の制限事項については、付録 C を参照してください。

#### 2.1.2.2 System V MNLS と Tru64 UNIX で名称が異なる関数を Tru64 UNIX の関数に置き換えるマクロ

MNLS マイグレーション・キットでは次の System V MNLS の関数をマクロで提供します。

- `iswascii` , `isphonogram` , `isideogram` , `isenglish` , `isnumber` , `isspecial`

これらのマクロは System V MNLS の関数と互換です。V3.0 から提供されている新しい文字クラスを使用して実現しています。

- `wscat` , `wsncat` , `wscmp` , `wsncmp` , `wscopy` , `wsncpy` , `wslen` , `wschr` , `wsrchr` , `wspbrk` , `wsspn` , `wscspn` , `wstok`

これらのマクロは System V MNLS の関数と互換です。

#### 2.1.2.3 EUC コードセットの画面制御情報を取得するマクロ

MNLS マイグレーション・キットでは `getwidth` 関数で取得した情報と併せて画面の制御に使用できるマクロを提供します。提供される機能は次のとおりです。

- `CODESETNO`

`CODESETNO` マクロは指定されたワイド文字を検査し、日本語 EUC コードセットに対応して、0 ~ 3 を返します。`CODESETNO` マクロの仕様については、付録 C を参照してください。

#### 2.1.2.4 `gettext` 関数

`gettext` 関数は、XPG4 国際化インタフェースには存在しませんが、Tru64 UNIX では提供されています。関数の形式は同じなので、プログラムの修正は必要ありません。Tru64 UNIX の `gettext` 関数は、XPG4 メッセージ・カタログを使用します。

---

## 2.2 移行の手順

MNLS マイグレーション・キットを使用した、アプリケーションの国際化機能の移行の手順は次のとおりです。

### 1. プログラム修正規模の把握

プログラム移行支援ツールを利用し、アプリケーション・プログラムの修正規模を把握します。

## 2. プログラムの修正

プログラム移行支援ツールが出力するプログラムの修正箇所の情報をもとにプログラムを修正します。

## 3. プログラムのコンパイルおよびリンク

修正されたプログラムをコンパイルおよびリンクして実行モジュールを生成します。

## 4. メッセージ・ファイルの変換

System V MNLS のメッセージ・ソース・ファイルをメッセージ・ファイル移行支援ツールを利用して、XPG4 メッセージ・ソース・ファイルに変換します。

## 5. メッセージ・カタログの生成

手順 4 で生成された XPG4 メッセージ・ソース・ファイルを gencat コマンドにより、メッセージ・カタログを生成します。

## 6. 生成されたメッセージ・カタログの内容の確認

手順 5 で生成されたメッセージ・カタログを dspcat コマンドにより内容を確認します。

---

## 2.3 プログラム修正規模の把握

たとえば、対象ファイルが test.c の場合、次のように入力して下さい。

```
% /usr/i18n/bin/svr4/chkmnls test.c
```

これで、プログラムの修正規模が把握できます。

プログラム移行支援ツール (chkmnls) に関する詳しい説明については、付録 C を参照してください。

---

## 2.4 プログラムの修正

MNLS マイグレーション・キットで提供される機能だけでは、MNLS アプリケーションはコンパイルできません。プログラムの次のような修正が必要です。

- MNLS マイグレーション・キットで未対応の関数に関する修正
- 書式付き入出力関数におけるワイド文字制御の変換仕様の修正
- EUC コードセットに依存した処理に関する修正
- curses ライブラリ関数に関する修正

### 2.4.1 MNLS マイグレーション・キットで未対応の関数に関する修正

MNLS マイグレーション・キットでは次の関数には未対応です。

- `addsev` , `lfmt` , `pfmt` , `setlabel` , `vlfmt` , `vpfmt`

これらの関数は、System V 固有の記録機能および監視機能を利用するためのものです。この機能は Tru64 UNIX ではサポートしていませんが、代用の関数を利用することで一部の機能を実現することが可能です。代用できる関数として、`fprintf` 関数、`fmsg` 関数そして `syslog` 関数が考えられますが、ここでは `fprintf` 関数と `fmsg` 関数を使用した修正の例について説明します。

`fprintf` 関数を使用する場合

- 制限事項

標準のメッセージ書式での編集ができません。メッセージ機構を利用したメッセージの編集ができません。

- `addsev` , `setlabel`

これらの関数は使用できませんので、コメントにします。

- `lfmt` , `vlfmt` , `pfmt` , `vpfmt`

`fprintf` 関数に置き換えます。以下に具体的な例を示します。

修正前:

```
#include <pfmt.h>
char *filename = "sample.dat";
setlabel("LABEL");
flag = MM_STD | MM_GET | MM_ERROR | MM_CONSOLE |
      MM_HARD | MM_APPL;
lfmt(stream, flag, "msgfile:1:not found(%s)\n",
      filename);
lfmt(stream, flag | MM_ACTION, "create file(%s)\n",
      filename);
```

stream への出力:

```
"LABEL:ERROR: not found(sample.dat)"
"LABEL:TO FIX: create file(sample.dat)"
```

修正後:

```
/* #include <pfmt.h> */
char *filename = "sample.dat";
/* setlabel("LABEL"); */
/*
flag = MM_STD | MM_GET | MM_ERROR | MM_CONSOLE |
      MM_HARD | MM_APPL;
lfmt(stream, flag, "msgfile:1:not found(%s)\n",
      filename);
lfmt(stream, flag | MM_ACTION, "create file(%s)\n",
      filename);
*/
fprintf(stream, "LABEL:ERROR,HARD,APPL: not found (%s)\n", filename);
fprintf(stream, "LABEL:TO FIX: create file(%s)\n",
      filename);
```

stream への出力:

```
"LABEL:ERROR,HARD,APPL: not found(sample.dat)"
"LABEL:TO FIX: create file(sample.dat)"
```

#### fmtmsg 関数を使用する場合

- 制限事項

メッセージ機構を利用したメッセージの編集ができません。メッセージの出力先は、コンソールまたは標準エラー出力のみです。コンソールへの出力にはタイム・スタンプは付加されません。

- addsev

この関数は使用できませんのでコメントにしますが、代わりに環境変数 SEV\_LEVEL で追加の重大度を定義します。

- setlabel

この関数は使用できませんのでコメントにしますが、代わりに fmtmsg 関数の引数でラベルを定義します。

- lfmt , vlfmt , pfmt , vpfmt

fmtmsg 関数に置き換えます。以下に具体的な例を示します。

修正前:

```
#include <pfmt.h>
char *filename = "sample.dat";
setlabel("LABEL");
flag = MM_STD | MM_GET | MM_ERROR | MM_CONSOLE |
      MM_HARD | MM_APPL;
lfmt(stream, flag, "msgfile:1:not found(%s)\n",
      filename);
lfmt(stream, flag | MM_ACTION, "create file(%s)\n",
      filename);
```

stream への出力:

```
"LABEL:ERROR: not found(sample.dat)"
"LABEL:TO FIX: create file(sample.dat)"
```



修正後:

```
/* #include <pfmt.h> */
#include <fmtmsg.h>
char *filename = "sample.dat";
char msgtext[255 +1];
char acttext[255 +1];
/* setlabel("LABEL"); */
/*
flag = MM_STD | MM_GET | MM_ERROR | MM_CONSOLE |
      MM_HARD | MM_APPL;
lfmt(stream, flag, "msgfile:1:not found(%s)\n",
      filename);
lfmt(stream, flag | MM_ACTION, "create file(%s)\n",
      filename);
*/
sprintf(msgtext, "not found(%s)\n", filename);
sprintf(acttext, "create file(%s)\n", filename);
fmtmsg(MM_HARD | MM_APPL | MM_CONSOLE | MM_PRINT,
"LABEL", MM_ERROR, msgtext, acttext,
MM_NULLTAG);
```

標準エラー出力およびコンソール への出力:

```
"LABEL:ERROR: not found(sample.dat)"
"LABEL:TO FIX: create file(sample.dat)"
```

- setcat

この関数は、gettxt 関数をメッセージ・ファイル名を省略して呼び出す時に使用されます。したがって、gettxt 関数を呼び出す時にメッセージ・ファイル名を省略せずに呼び出すように書き換える必要があります。以下に具体的な例を示します。

修正前:

```
setcat ("msgfile");
msg = gettxt (":10", "default msg");
```

修正後:

```
msg = gettxt ("msgfile:10", "default msg");
```

### 2.4.2 書式付き入出力関数におけるワイド文字制御の変換仕様の修正

書式付き入出力関数で使用する、制御文字列内のワイド文字を制御する変換仕様は、Tru64 UNIX と System V MNLS では次のように異なります。

- Tru64 UNIX

ワイド文字は %C、ワイド文字列は %S が使用されます。

- System V MNLS

ワイド文字は %wc、ワイド文字列は %ws が使用されます。

したがって、アプリケーションで使用しているこれらの変換仕様 (%wc %ws) を Tru64 UNIX の変換仕様 (%C %S) に書き換える必要があります。

コンパイル時には、この機能の違いは見つかりませんので注意が必要です。

また、ここでいう書式付き入出力関数とは次の関数を指します。

- printf, fprintf, sprintf, vprintf, vfprintf, vsprintf
- scanf, fscanf, sscanf

### 2.4.3 EUC コードセットに依存した処理に関する修正

System V MNLS の場合、使用できるコードセットが EUC コードセットに限定されているため、たとえば EUC コードセットのビット・パターンを利用して、文字の分類をするということが可能です。しかし、EUC コードセット以外のコードセットの場合この方法は使用できません。このビット・パターンを利用したプログラムの記述とは、たとえば、ある文字が ASCII コードセットかどうかの判定をする場合に、0x80 と論理積 (AND) をとるといったことです。このような EUC コードセットに依存したプログラムの記述はすべて排除する必要があります。

#### 2.4.4 curses ライブラリ関数に関する修正

V4.0 より前のバージョンのオペレーティング・システムでは、国際化された curses ライブラリに関して次のような修正が必要です。

- pechowchar 関数を次の関数の組み合わせの内、いずれかで置き換えます。
  - (1) addwch 関数, refresh 関数
  - (2) waddwch 関数, wrefresh 関数
  - (3) waddwch 関数, prefresh 関数
- 次の関数は文字数の指定に 0 または, -1 以外の負の整数を指定している場合は, -1 を指定するようにします ( Tru64 UNIX の curses ライブラリでは, -1 を指定すると文字列全体を指定したことになります)。  
`insnwstr, winsnwstr, mvinsnwstr, mvwinsnwtr`

---

### 2.5 プログラムのコンパイルおよびリンク

MNLS マイグレーション・キットで提供されるヘッダ・ファイルとライブラリを使用するには次の作業が必要です。

- コンパイル・オプションに MNLS マイグレーション・キットのヘッダ・ファイルの格納先を指定する
- リンク・オプションに MNLS マイグレーション・キットのライブラリの格納先を指定する

#### 2.5.1 コンパイル・オプションに MNLS マイグレーション・キットのヘッダ・ファイルの格納先を指定する

コンパイル・オプションに MNLS マイグレーション・キットのヘッダ・ファイルの格納先を次のように指定します。

```
-I/usr/i18n/include/svr4
```

たとえば、次のように入力してください。

```
% cc -c -I/usr/i18n/include/avr4 main.c more.c
```

プログラムのコンパイルに関する詳しい説明については、『Tru64 UNIX プログラミング・ガイド』を参照してください。

### 2.5.2 リンク・オプションに MNLS マイグレーション・キットのライブラリの格納先を指定する

リンク・オプションに MNLS マイグレーション・キットのライブラリの格納先を次のように指定します。

```
-L/usr/i18n/lib/avr4
```

たとえば、次のように入力してください。

```
% cc -c -I/usr/i18n/include/avr4 main.c more.c  
% cc -o more main.o more.o -L/usr/i18n/lib/avr4 -lw
```

プログラムのリンクに関する詳しい説明については、『Tru64 UNIX プログラミング・ガイド』を参照してください。

---

## 2.6 メッセージ・ファイルの変換

たとえば、System V MNLS のメッセージ・ソース・ファイルが msg.txt、変換後の XPG4 メッセージ・ソース・ファイルが msg\_xpg4.msg の場合、次のように入力して下さい。

```
% /usr/i18n/bin/avr4/msgtrn msg.txt msg_xpg4.msg
```

これで、XPG4 メッセージ・ソース・ファイル msg\_xpg4.msg が生成されます。メッセージ・ファイル移行支援ツールについての詳しい説明については、付録 C を参照してください。

---

## 2.7 メッセージ・カタログの生成

たとえば、生成された XPG4 メッセージ・ソース・ファイルが `msg_xpg4.msg`、生成する XPG4 メッセージ・カタログが `msg_xpg4.cat` の場合、次のように入力して下さい。

```
% gencat msg_xpg4.cat msg_xpg4.msg
```

これで、XPG4 メッセージ・カタログ `msg_xpg4.cat` が生成されます。

---

## 2.8 生成されたメッセージ・カタログの内容の確認

たとえば、生成された XPG4 メッセージ・カタログが `msg_xpg4.cat` の場合、次のように入力して下さい。

```
% dspcat msg_xpg4.cat
```

これで、メッセージ・カタログ `msg_xpg4.cat` の内容を確認することができます。



---

## XP4 国際化インタフェースを使用した国際化機能の移行

---

### 3.1 国際化機能移行支援ツール

MNLS マイグレーション・キットを使用する場合は、比較的少ない修正で済みます。しかし、プログラムのポータビリティのために XP4 国際化インタフェースを使用して移行を行う場合は、直接アプリケーションを修正する必要があります。これはとても大がかりな作業です。そこで、日本語 Tru64 UNIX では、この移行作業を軽減するために次のツールを提供しています。

- プログラム移行支援ツール

プログラム中の System V MNLS の関数名やヘッダ・ファイル名の検索を行い、プログラムの修正の難易度や修正の必要な箇所などの情報を出力します。

- メッセージ・ファイル移行支援ツール

System V MNLS のメッセージ・ソース・ファイル (テキスト形式) を XP4 メッセージ・ソース・ファイル (テキスト形式) に変換するツールです。

---

### 3.2 移行の手順

国際化機能移行支援ツールを使用した、アプリケーションの国際化機能の移行の手順は次のとおりです。

1. プログラム修正規模の把握

プログラム移行支援ツールを利用し、アプリケーション・プログラムの修正規模を把握します。

2. プログラムの修正

プログラム移行支援ツールが出力するプログラムの修正箇所の情報をもとにプログラムを修正します。

3. プログラムのコンパイルおよびリンク

修正されたプログラムをコンパイルおよびリンクして実行モジュールを生成します。

4. メッセージ・ファイルの変換

System V MNLS のメッセージ・ソース・ファイルをメッセージ・ファイル移行支援ツールを利用して、XPG4 メッセージ・ソース・ファイルに変換します。

5. メッセージ・カタログの生成

手順 4 で生成された XPG4 メッセージ・ソース・ファイルを `gencat` コマンドにより、メッセージ・カタログを生成します。

6. 生成されたメッセージ・カタログの内容の確認

手順 5 で生成されたメッセージ・カタログを `dspcat` コマンドにより内容を確認します。

---

### 3.3 プログラム修正規模の把握

たとえば、対象ファイルが `test.c` の場合、次のように入力して下さい。

```
% /usr/i18n/bin/svr4/chkmnls test.c
```

これで、プログラムの修正規模が把握できます。

---

### 3.4 プログラムの修正

たとえば、対象ファイルが `test.c` の場合、次のように入力して下さい。

```
% /usr/i18n/bin/svr4/chkmnls -m test.c
```



これで、プログラムの修正の必要な箇所にコメントが付加されたファイル `test.c.chk` が生成されます。この生成されたファイル `test.c.chk` を参考に `test.c` を修正します。プログラム移行支援ツールについての詳しい説明は付録 C を、プログラムの修正に関しての詳しい説明は、付録 A を参照してください。

---

### 3.5 プログラムのコンパイル

MNLS マイグレーション・キットを使用した国際化機能の移行と違い、修正後のプログラムは Tru64 UNIX でのコンパイルおよびリンクには、特別な指定は必要ありません。たとえば、対象ファイルが `test.c` の場合、次のように入力してください。

```
% cc test.c
```

---

### 3.6 メッセージ・ファイルの変換

たとえば、System V MNLS のメッセージ・ソース・ファイルが `msg.txt`、変換後の XPg4 メッセージ・ソース・ファイルが `msg_xpg4.msg` の場合、次のように入力して下さい。

```
% /usr/i18n/bin/svr4/msgtrn msg.txt msg_xpg4.msg
```

これで、XPg4 メッセージ・ソース・ファイル `msg_xpg4.msg` が生成されます。メッセージ・ファイル移行支援ツールについての詳しい説明については、付録 C を参照してください。

---

### 3.7 メッセージ・カタログの生成

たとえば、XPg4 メッセージ・ソース・ファイルが `msg_xpg4.msg`、生成する XPg4 メッセージ・カタログが `msg_xpg4.cat` の場合、次のように入力して下さい。

```
% gencat msg_xpg4.cat msg_xpg4.msg
```

これで、XPG4 メッセージ・カタログ `msg_xpg4.cat` が生成されます。

---

### 3.8 生成されたメッセージ・カタログの内容の確認

たとえば、生成された XPG4 メッセージ・カタログが `msg_xpg4.cat` の場合、次のように入力して下さい。

```
% dspcat msg_xpg4.cat
```

---

## MNLS から XPG4 への国際化機能移行ガイド

この章では、MNLS アプリケーションを XPG4 国際化インタフェースを使用した記述に書き換える際の修正の方法や、プログラムの実行時に必要な設定について説明します。

---

### A.1 プログラムの修正方法

System V MNLS ライブラリ関数と XPG4 の国際化インタフェースでは、ライブラリの関数名やヘッダ・ファイル名が違っている部分があり、プログラムを修正する必要があります。この節では、ヘッダ・ファイルの違いやライブラリの関数名の違いなどで必要となる修正方法について説明します。

#### A.1.1 ヘッダ・ファイル

System V MNLS のヘッダ・ファイルと XPG4 の国際化インタフェースのヘッダ・ファイルを比較したのが次の表です。

表 A-1 ヘッダ・ファイル比較一覧

MNLS	XPG4
ctype.h	
getwidth.h	-
locale.h	
pfmt.h	-

(次ページに続く)

表 A-1 (続き) ヘッダ・ファイル比較一覧

MNLS	XPG4
stdarg.h	
stdio.h	(unistd.h) <sup>1</sup>
stdlib.h	
sys/euc.h	-
wctype.h	wchar.h
widec.h	wchar.h

<sup>1</sup> getopt 関数のプロトタイプ宣言については、System V MNLS では stdio.h ですが、XPG4 では unistd.h に変更されています。

この表の意味とプログラムの修正方法は次のとおりです。

- XPG4 の欄に - があるもの  
System V MNLS 固有のヘッダ・ファイルです。プログラムから削除（または、コメントに）する必要があります。
- XPG4 の欄にヘッダ・ファイル名があるもの  
System V MNLS と XPG4 国際化インタフェースでファイル名が異なるものです。プログラムのヘッダ・ファイルの展開箇所を修正する必要があります。
- XPG4 の欄に があるもの  
System V MNLS と XPG4 の国際化インタフェースでヘッダ・ファイルのファイル名が一致しているものです。プログラムの修正は必要ありません。

### A.1.2 ライブラリ関数

System V MNLS のライブラリ関数と XPG4 の国際化インタフェースの比較をしたのが次の表です。

表 A-2 ライブラリ関数比較一覧

MNLS	XPG4	type
addsev	-	-
fgetwc		×
fgetws		
fprintf		
fputwc		×
fputws		
fscanf		
getopt		
gettxt	-	-
getwc		×
getwchar		×
getwidth	-	-
getws	-	-
isenglish	-	-
isideogram	-	-
isnumber	-	-
isphonogram	-	-
isspecial	-	-
iswalnum		
iswalpha		
iswascii	-	-
iswcntrl		
iswdigit		
iswgraph		
iswlower		
iswprint		
iswpunct		
iswspace		

(次ページに続く)

表 A-2 (続き) ライブラリ関数比較一覧

MNLS	XPG4	type
iswupper		
iswxdigit		
lfmt	-	-
mblen		
mbstowcs		
mbtowc		
pfmt	-	-
printf		
putwc		×
putwchar		×
putws	-	-
scanf		
setcat	-	-
setlabel	-	-
setlocale		
sprintf		
sscanf		
strtows	-	-
towlower		
towupper		
ungetwc		×
vfprintf		
vlfmt	-	-
vpfmt	-	-
vprintf		
vsprintf		
wcstombs		
wctomb		

(次ページに続く)

表 A-2 (続き) ライブラリ関数比較一覧

MNLS	XPG4	type
wscat	wscat	
wschr	wschr	×
wscmp	wscmp	
wscpy	wscpy	
wscspn	wscspn	×
wslen	wslen	×
wsncat	wsncat	×
wsncmp	wsncmp	×
wsncpy	wsncpy	×
wspbrk	wspbrk	
wsrchr	wsrchr	×
wsspnp	wsspnp	×
wstok	wstok	
wstostr	-	-

この表の意味とプログラムの修正要領は次のとおりです。

- XPG4 の欄に - があるもの

System V MNLS 固有の関数です。プログラムを修正して同等の機能を実現する必要があります。

- XPG4 の欄に関数名があるもの

System V MNLS と XPG4 国際化インタフェースで関数名が異なるものです。機能的な差がありませんから、直接関数名を書き換えるか、マクロによる置き換えが必要です。type の欄が になっているものは、関数名を書き換える以外にプログラムの修正は必要ありません。

- XPG4 の欄に があるもの

System V MNLS と XPG4 の国際化インタフェースで関数名が一致しているものです。type の欄が になっているものは、プログラムの修正は必要ありません。

- type の欄に × があるもの

引数の型や、リターン値の型が違っているものです。関数の呼び出し時の引数の型やリターン値の型を修正する必要があります。以下に具体的に例を示します。

型の違いを修正した例 (リターン値)

– MNLS

```
int i;  
i = getwchar();          /* リターン値の型が違います */
```

– XPG4

```
wint_t i;                /* int -> wint_t */  
i = getwchar();
```

• type の欄に があるもの

引数の型や、リターン値の型が一致しているものです。wchar\_t 型と wint\_t 型の違いのみの関数は、type 欄が になっています。

A.1.2.1 System V MNLS 固有の関数の対応について

XPG4 国際化インタフェースにない System V MNLS 固有の関数に対する修正は、次のとおりです。

• getwidth

この関数は、EUC コードセットに強く依存しており、直接 XPG4 国際化インタフェースで置き換えることができません。代替りとして、文字や文字列の画面上での表示幅を取得するには、wcwidth 関数や wcswidth 関数を使用してください。

• addsev, lfmt, pfmt, setlabel, vlfmt, vpfmt

これらの関数は、System V 固有の記録機能および監視機能を利用するためのものです。fprintf 関数を利用することで一部の機能を実現することが考えられます。ここでは fprintf 関数を利用した修正の例について説明します。

• 制限事項

標準のメッセージ書式での編集ができません。メッセージ機構を利用したメッセージの編集ができません。



- `lfmt` , `vlfmt` , `pfmt` , `vpfmt`

`fprintf` 関数に置き換えます。以下に具体的な例を示します。

修正前:

```
#include <pfmt.h>
char *filename = "sample.dat";
setlabel("LABEL");
flag = MM_STD | MM_GET | MM_ERROR | MM_CONSOLE |
      MM_HARD | MM_APPL;
lfmt(stream, flag, "msgfile:1:not found(%s)\n",
      filename);
lfmt(stream, flag | MM_ACTION, "create file(%s)\n",
      filename);
```

stream への出力:

```
"LABEL:ERROR: not found(sample.dat)"
"LABEL:TO FIX: create file(sample.dat)"
```

修正後:

```
/* #include <pfmt.h> */
char *filename = "sample.dat";
/* setlabel("LABEL"); */
/*
flag = MM_STD | MM_GET | MM_ERROR | MM_CONSOLE |
      MM_HARD | MM_APPL;
lfmt(stream, flag, "msgfile:1:not found(%s)\n",
      filename);
lfmt(stream, flag | MM_ACTION, "create file(%s)\n",
      filename);
*/
fprintf(stream, "LABEL:ERROR,HARD,APPL:not found(%s)\n",
      filename);
fprintf(stream, "LABEL:TO FIX: create file(%s)\n",
      filename);
```

stream への出力:

```
"LABEL:ERROR,HARD,APPL: not found(sample.dat)"
"LABEL:TO FIX: create file(sample.dat)"
```

- `addsev` , `setlabel`

これらの関数は使用できませんので、コメントにします。

- `gettext` , `setcat`

`gettext` 関数の代わりには、`catopen` 関数と `catgets` 関数および `catclose` 関数を使用してください。また、`setcat` 関数で指定していたメッセージ・ファイルは `catopen` 関数の呼び出しの時に使用してください。以下に具体的な例を示します。

修正前:

```
char *msg;

setcat ("msgfile");
msg = gettext (":10", "default msg");
printf("%s\n", msg);
```

修正後:

```
char *msg;
nl_catd catd;

catd = catopen("msgfile", NL_CAT_LOCALE);
msg = catgets(catd, 1, 10, "default msg");
printf("%s\n", msg);
catclose(catd);
```

- `getws`

この関数に対しては次の二つの方法があります。

- `fgetws` 関数を利用する

`fgetws` に置き換える方法です。`fgetws` 関数の引数には文字列の格納先、`stdin`、文字列の格納先の配列のサイズを指定します。ただし、`fgetws` 関数は改行文字がそのまま配列に取り込まれるので注意が必要です。

- `getwchar` 関数を利用して、`getws` 関数に代わる関数を用意する

`fgetws` 関数を利用する場合と違い、この場合は新たに関数を作成することになります。プログラムの `getws` 関数を呼び出している箇所をその新しい関数に書き換えます。

- `putws`

`putws` 関数を `fputws` 関数に書き換えます。 `fputws` 関数の引数には文字列の格納先、 `stdin` を指定します。ただし、 `fputws` 関数は改行文字を出力しませんから、 `putwchar` 関数の引数に改行文字を指定して呼び出し、改行文字を出力する必要があります。また、このような処理を一つの関数にして、その関数を呼び出すように書き換える方法もあります。

- `isenglish` , `isideogram` , `isnumber` , `isphonogram` , `isspecial` , `iswascii`

これらの関数に対応する文字クラスを XPG4 では提供していませんから、新たに同等の文字クラスを作成して `iswctype` 関数を呼び出すか、プログラムのロジックを書き直す必要があります。 `iswctype` 関数の使用方法については、 `/usr/include/avr4/wctype.h` を参考にすると良いでしょう。

- `strtows`

`mbstowcs` 関数で置き換えることができます。ただし、 `mbstowcs` 関数の引数に格納後の文字列の文字数を指定するために、扱える文字列の長さに制限が発生します。また、 `mbstowcs` 関数はポインタではなく、処理した文字数を返すため、リターン値の判定処理を修正する必要があります。そこで、 `strtows` 関数と同じようにポインタを返す関数を `mbstowcs` 関数を使用して作成し、その関数を呼び出すように書き換えるのが良いでしょう。

- `wstostr`

`wcstombs` 関数で置き換えることができます。ただし、 `wcstombs` 関数の引数に格納後の文字列のバイト数を指定するために、扱える文字列の長さに制限が発生します。また、 `wcstombs` 関数はポインタではなく、処理したバイト数を返すため、リターン値の判定処理を修正する必要があります。そこで、 `wstostr` 関数と同じようにポインタを返す関数を `wcstombs` 関数を使用して作成し、その関数を呼び出すように書き換えるのが良いでしょう。

#### A.1.2.2 書式付き入出力関数におけるワイド文字制御の変換仕様

書式付き入出力関数で使用する、制御文字列内のワイド文字を制御する変換仕様は、 System V MNLS と XPG4 国際化インタフェースでは次のように異なります。

- System V MNLS

ワイド文字は `%wc` , ワイド文字列は `%ws` が使用されます。

- XPG4 国際化インタフェース

ワイド文字は %C, ワイド文字列は %S が使用されます。

したがって、アプリケーションで使用しているこれらの変換仕様 (%wc %ws) を XPG4 国際化インタフェースの変換仕様 (%C %S) に書き換える必要があります。

コンパイル時には、この機能の違いは見つかりませんので注意が必要です。

また、ここでいう書式付き入出力関数とは次の関数を指します。

- printf, fprintf, sprintf, vprintf, vfprintf, vsprintf
- scanf, fscanf, sscanf

#### A.1.3 EUC コードセットに依存した処理

System V MNLS の場合、使用できるコードセットが EUC コードセットに限定されているため、たとえば EUC コードセットのビット・パターンを利用して、文字の分類をするということが可能です。しかし、EUC コードセット以外のコードセットの場合この方法は使用できません。このビット・パターンを利用したプログラムの記述とは、たとえば、ある文字が ASCII コードセットかどうかの判定をする場合に、0x80 と論理積 (AND) を取るといったことです。このような EUC コードセットに依存したプログラムの記述はすべて排除する必要があります。

#### A.1.4 curses ライブラリ関数

国際化された curses ライブラリは、XPG4 では規定されていません。プログラムを移行する場合は、移行先の OS に依存することになります。Tru64 UNIX での移行については、第 2 章を参照して下さい。

---

## A.2 メッセージ・ファイルの移行

System V MNLS のメッセージ・ソース・ファイルはメッセージ・ファイル移行支援ツールで XPG4 メッセージ・ソース・ファイルに変換できます。メッセージ・ファイル移行支援ツールについての詳しい説明については、付録 C を参照してください。

---

## A.3 プログラム実行時の設定

System V MNLS ライブラリ関数と XPG4 の国際化インタフェースとでは、プログラムの実行時の設定に次のような違いがあります。

- メッセージ機構で使用するメッセージ・ファイルの格納場所

System V MNLS では、`/usr/lib/locale` の下の特定のディレクトリに格納しますが、XPG4 では、`NLSPATH` 環境変数で指定すればどのディレクトリでも格納できます。環境変数 `NLSPATH` の設定についての詳しい説明については、『国際化ソフトウェア・プログラミング・ガイド』を参照してください。



# B

## プログラムの移行例

この章では、Tru64 UNIX のディレクトリ/usr/i18n/examples/mnls\_xpg4 に格納されているサンプル・プログラムを例に取り、MNLS アプリケーションの日本語 Tru64 UNIX への移行を MNLS マイグレーション・キットを用いた場合と、XPG4 国際化インタフェースへの書き換えによる場合とに分けて説明します。

### B.1 サンプル・プログラム

Tru64 UNIX のディレクトリ/usr/i18n/examples/mnls\_xpg4 には、次のようなサンプル・プログラムが格納されています。

- ディレクトリ構成

表 B-1 ディレクトリ構成

ディレクトリ名	内容
MNLS	修正前の MNLS 用のアプリケーション・プログラム
OSF1	上記のディレクトリ MNLS に格納されたアプリケーション・プログラムを MNLS マイグレーション・キットを利用して、修正したものが格納されています。
XPG4	上記のディレクトリ MNLS に格納されたアプリケーション・プログラムを XPG4 国際化インタフェースを利用して、修正したものが格納されています。

- 機能

テキスト・ファイルから単語を読み込み、単語を構成する文字により次の表のように分類して標準出力に出力します。

表 B-2 分類

表音文字:	全角のひらがな, 全角のカタカナ
表意文字:	漢字
句読文字:	! " # \$ % & ' ( ) * + , - . / ; < = > ? @ [ ¥ ] ^ _ ' {   } ~ ....
英小文字:	a b c ... x y z , a b c ... x y z
英大文字:	A B C ... X Y Z , A B C ... X Y Z
数字:	0 1 2 ... 7 8 9 , 0 1 2 ... 7 8 9

単語が複数の分類に属する場合は, この表の上の項目の方に分類されます。

読み込むテキスト・ファイルの形式は, 各単語が ':' (コロン) で区切られたもので, 単語の最大は 16 文字まで, 行の最大は 80 文字までです。ファイル名は mnlsdemo.dat で, 固定です。ファイルはカレントディレクトリに置かれている必要があります。

- プログラムのファイル構成

ファイル名	処理内容
mnlsdemo.h	構造体の定義, 固定値の定義, 関数のプロトタイプ宣言
mnlsdemo_main.c	メイン・コントロール処理, データ・ファイルのオープン, クローズ
mnlsdemo_getword.c	単語読み込み処理
mnlsdemo_class.c	分類処理
mnlsdemo_disp.c	表示処理
mnlsdemo_msg.txt	メッセージ・ソース・ファイル (MNLS)
mnlsdemo_msg.msg	メッセージ・ソース・ファイル (OSF/1, XPG4)
mnlsdemo.dat	サンプル・データ。ディレクトリ MNLS には, 入っていません。
Makefile	プログラムのコンパイル, メッセージ・カタログ生成用の make コマンド用スクリプト。ディレクトリ MNLS には, 入っていません。
mnlsdemo_msg.cat	XPG4 メッセージ・カタログ。make コマンドにより上記 Makefile を実行すると生成されます。



B.2

MNLS マイグレーション・キットを利用した移行例

ディレクトリ MNLS に格納されたプログラムを MNLS マイグレーション・キットを利用して、国際化機能の移行するにはどのような修正や設定が必要かを説明します。修正された結果は、ディレクトリ OSF1 に格納されています。

B.2.1

プログラム移行支援ツールによる修正箇所の調査

MNLS マイグレーション・キットで対応していない System V MNLS の機能を使用している箇所は修正が必要です。どこを修正すればよいかを検出するために、プログラム移行支援ツールを使用します。次のように入力して下さい。

```
% cd /usr/i18n/examples/mnls_xpg4/MNLS
% /usr/i18n/bin/svr4/chkmnls *.ch
```

これで、どのファイルで System V MNLS の機能を使っているかが把握できます。ただし、プログラム移行支援ツールは、MNLS マイグレーション・キットで対応している機能も検出しますので注意してください。この出力結果から、MNLS マイグレーション・キットで対応していない System V MNLS の機能を使用しているのは次のファイルです。

ファイル名	System V MNLS の機能
mnlsdemo_main.c	setcat
mnlsdemo_disp.c	%ws

実際は、これとは別に、プログラム移行支援ツールの -m オプションを使用して、修正箇所にコメントが付加されたファイルを生成し、それをもとに修正箇所を調査するとより正確になるのですが、ここでは省略します。

## B.2.2 プログラム移行支援ツールで検出できない修正箇所の調査

プログラム移行支援ツールは、System V MNLS の関数名やヘッダ・ファイル名を検出しますから、それ以外の EUC コードセットに依存したプログラムなどの記述は検出できません。したがって、直接プログラムを調査する必要があります。サンプル・プログラムを全体的に見直してみると、次のようにワイド文字のビット・パターンをチェックしている箇所などがあります。

ファイル名	System V MNLS の機能
mnlsdemo_disp.c	ワイド文字のビット・パターンでコードセットを判別
mnlsdemo.h	ワイド文字のビット・パターン判定用のビット・マスク System V MNLS のメッセージ機構用のメッセージ・ファイル System V MNLS のメッセージ機構用のメッセージ番号

## B.2.3 プログラムの修正

これまでの調査から、サンプル・プログラムの修正箇所は次のようになります。

- mnlsdemo.h

関数 setcat は、MNLS マイグレーション・キットでは対応していないので、次のような修正が必要です。

修正前:

```
#define MSGID_PHONO      ":1"
#define MSGID_IDEO      ":2"
#define MSGID_PUNCT      ":3"
#define MSGID_LOWER      ":4"
#define MSGID_UPPER      ":5"
#define MSGID_NUMERIC    ":6"
#define MSGID_OTHERS     ":7"
#define MSGID_ERROPEN    ":8"
```

修正後:

```
#define MSGID_PHONO          "mnlsdemo_msg.cat:1"
#define MSGID_IDEO           "mnlsdemo_msg.cat:2"
#define MSGID_PUNCT          "mnlsdemo_msg.cat:3"
#define MSGID_LOWER          "mnlsdemo_msg.cat:4"
#define MSGID_UPPER          "mnlsdemo_msg.cat:5"
#define MSGID_NUMERIC        "mnlsdemo_msg.cat:6"
#define MSGID_OTHERS         "mnlsdemo_msg.cat:7"
#define MSGID_ERROPEN        "mnlsdemo_msg.cat:8"
```

MNLS メッセージ・ファイルを XPG4 メッセージ・カタログに変換するので、ファイル名を修正します。

修正前:

```
#define MSGFILENAME  "mnlsdemo_msg"
```

修正後:

```
#define MSGFILENAME  "mnlsdemo_msg.cat"
```

ワイド文字のビット・パターン判定用のビット・マスクがありますので、次のようにコメントにします。

```
/*
#define MSKCHK_CODESET_0          (wchar_t)0x00000000
#define MSKCHK_CODESET_1          (wchar_t)0x30000000
#define MSKCHK_CODESET_2          (wchar_t)0x10000000
#define MSKCHK_CODESET_3          (wchar_t)0x20000000
#define MSK_CODESET               (wchar_t)0xf0000000
*/
```

- **mnlsdemo\_main.c**

関数 `setcat` は、MNLS マイグレーション・キットでは対応していないので、次のようにコメントにします。

```
/* setcat(MSGFILENAME); */
```

- **mnlsdemo\_disp.c**

ワイド文字のビット・パターンにより文字列の表示幅を処理していますので、次のように `wcswidth` 関数に置き換えます。

プログラムの移行例  
B.2 MNLS マイグレーション・キットを利用した移行例

修正前:

```
dsp_size =  
    mnlsdemo_getdspwidth(wbuf->wkind_buf[kind]col,  
                          widthset );
```

修正後:

```
dsp_size =  
    wcswidth(wbuf->wkind_buf[kind]col,  
             wcslen(wbuf->wkind_buf[kind]col));
```

**mnlsdemo\_getdspwidth** 関数が不要になりますので、関数全体をコメントにします。

```
#ifdef MNLS_UNDELETE  
int mnlsdemo_getdspwidth(wchar_t *wkind_buf, eucwidth_t *widthset)  
{  
    .  
    .  
    この間には、関数の本体の記述があります。  
    .  
    .  
}  
#endif MNLS_UNDELETE
```

書式付き出力関数でのワイド文字制御の変換仕様は (%ws %wc) は、MNLS マイグレーション・キットでは対応していないので、次のように修正します。

修正前:

```
printf("%ws\n", wln_buf);
```

修正後:

```
printf("%S\n", wln_buf);
```

## B.2.4 メッセージ・ファイルの変換

System V MNLS 用に作成されたメッセージ・ソース・ファイルをメッセージ・ファイル移行支援ツールを用いて、XPG4 メッセージ・ソース・ファイルに変換します。次のように入力すると変換された XPG4 メッセージ・ソース・ファイル `mnlsdemo_msg.msg` がホーム・ディレクトリに生成されます。

```
% cp /usr/i18n/examples/mnls_xpg4/MNLS/mnlsdemo_msg.txt $HOME/mnlsdemo_msg.org
% /usr/i18n/bin/svr4/msgtrn $HOME/mnlsdemo_msg.org mnlsdemo_msg.msg
```

## B.2.5 プログラムのコンパイル

これまでの修正の結果が、`/usr/i18n/examples/mnls_xpg4/OSF1` に格納されています。これをコンパイルするには、たとえば次のようにしてホーム・ディレクトリにコピーしてから行って下さい。

```
% cp -r /usr/i18n/examples/mnls_xpg4/OSF1 $HOME
% cd $HOME/OSF1
% make all
% make messages
```

これにより、プログラムの実行モジュールおよびメッセージ・カタログが生成されます。

## B.2.6 プログラムの実行

生成された実行モジュールを実行するには、次の環境変数の設定が必要です。

- NLSPATH

メッセージ・カタログのサーチ・パスを設定します。サンプルのメッセージ・カタログは、プログラムと同じディレクトリ (`$HOME/OSF1`) に生成されていますから、次のように設定します。

```
% setenv NLSPATH $HOME/OSF1/%N:$NLSPATH
```

- LANG

## プログラムの移行例

### B.2 MNLS マイグレーション・キットを利用した移行例

サンプルのメッセージやデータ・ファイルは ja\_JP.eucJP ロケールを使用しているので、次のように設定します。

```
% setenv LANG ja_JP.eucJP
```

- CSWIDTH

マイグレーション・キットの getwidth 関数が使われます。ja\_JP.eucJP ロケールでは、次のように設定します。

```
setenv CSWIDTH 2:2,1:1,2:2
```

以上の設定で実行モジュールを実行します。

---

## B.3 XPG4 国際化インタフェースへの書き換えによる移行例

ディレクトリ MNLS に格納されたプログラムを XPG4 国際化インタフェースを利用して、国際化機能の移行をするにはどのような修正や設定が必要かを説明します。修正された結果は、ディレクトリ XPG4 に格納されています。

### B.3.1 プログラム移行支援ツールによる修正箇所の調査

どこを修正すればよいかを検出するために、プログラム移行支援ツールを使用します。次のように入力して下さい。

```
% cd /usr/i18n/examples/mnls_xpg4/MNLS
% /usr/i18n/bin/svr4/chkmnls *. [ch]
```

これで、どこで System V MNLS の機能を使っているかが把握できます。

MNLS の機能を使用しているのは次のファイルです。

ファイル名	System V MNLS の機能
mnlsdemoe.h	<sys/euc.h> <wdec.h> eucwidth_t

ファイル名	System V MNLS の機能
mnlsdemo_main.c	<getwidth.h> gettxt getwidth setcat wscpy
mnlsdemo_getword.c	wscpy wslen wstok
mnlsdemo_class.c	<wctype.h> <wdec.h> isphonogram isnumber isideogram wslen
mnlsdemo_disp.c	eucwidth_t gettxt wscat wslen %ws

B.3.2    プログラム移行支援ツールで検出できない修正箇所の調査

プログラム移行支援ツールは、System V MNLS の関数名やヘッダ・ファイル名を検出しますから、それ以外の EUC コードセットに依存したプログラムなどの記述は検出できません。したがって、直接プログラムのロジックを調査する必要があります。サンプル・プログラムを全体的に見直してみると、次のようにワイド文字のビット・パターンをチェックしている箇所などがあります。

ファイル名	System V MNLS の機能
mnlsdemo_disp.c	ワイド文字のビット・パターンでコードセットを判別
mnlsdemo.h	ワイド文字のビット・パターン判定用のビット・マスク System V MNLS のメッセージ機構用のメッセージ・ファイル System V MNLS のメッセージ機構用のメッセージ番号

## プログラムの移行例

### B.3 XPG4 国際化インタフェースへの書き換えによる移行例

#### B.3.3 プログラムの修正

XPG4 国際化インタフェースを利用して、国際化機能の移行をする場合は、MNLS マイグレーション・キットを利用する場合と違い、修正箇所が多いですから、プログラム移行支援ツールの -m オプションを指定して、コメントの付加されたプログラムを見ながら修正します。たとえば次のようにしてホーム・ディレクトリにコピーしてから行って下さい。

```
% cp -r /usr/i18n/examples/mnls_xpg4/MNLS $HOME
% cd $HOME/MNLS
% /usr/i18n/bin/svr4/chkmnls -m *.hc]
```

これにより、ファイル名に.chk が付加されたファイルが生成されます。この生成されたファイルを参照しながら、プログラムを修正します。

- mnlsdemo.h

ヘッダ・ファイルを XPG4 国際化インタフェースに書き換えます。

修正前:

```
#include <sys/euc>
#include <wdec.h>
```

修正後:

```
/* #include <sys/euc> */
/* #include <wdec.h> */
#include <wchar.h>
#include <nl_types.h>
```

<nl\_types.h>は、catopen 関数、catgets 関数、catclose 関数を使用する時に必要です。

MNLS メッセージ・ファイルを XPG4 メッセージ・カタログに変換するので、ファイル名を修正します。

修正前:

```
#define MSGFILENAME          "mnlsdemo_msg"
```



修正後:

```
#define MSGFILENAME          "mnlstdemo_msg.cat"
```

メッセージ番号を `catgets` 関数の引数として利用できるように変更します。

修正前:

```
#define MSGID_PHONO          ":1"  
#define MSGID_IDEO          ":2"  
#define MSGID_PUNCT         ":3"  
#define MSGID_LOWER         ":4"  
#define MSGID_UPPER         ":5"  
#define MSGID_NUMERIC       ":6"  
#define MSGID_OTHERS        ":7"  
#define MSGID_ERROPEN       ":8"
```

修正後:

```
#define MSGID_PHONO          1  
#define MSGID_IDEO          2  
#define MSGID_PUNCT         3  
#define MSGID_LOWER         4  
#define MSGID_UPPER         5  
#define MSGID_NUMERIC       6  
#define MSGID_OTHERS        7  
#define MSGID_ERROPEN       8
```

ワイド文字のビット・パターン判定用のビット・マスクがありますので、次のようにコメントにします。

```
/*  
#define MSKCHK_CODESET_0      (wchar_t)0x00000000  
#define MSKCHK_CODESET_1      (wchar_t)0x30000000  
#define MSKCHK_CODESET_2      (wchar_t)0x10000000  
#define MSKCHK_CODESET_3      (wchar_t)0x20000000  
#define MSK_CODESET           (wchar_t)0xf0000000  
*/
```

`mnlstdemo_disp` 関数では、`gettext` 関数を使用しているため、`catgets` 関数に置き換えますが、引数のカタログ記述子を `main` 関数から引き継ぐ必要があるため、引数を変更します。

## プログラムの移行例

### B.3 XPG4 国際化インタフェースへの書き換えによる移行例

修正前:

```
extern void mnlsdemo_disp    (struct wkindarry *, eucwidth_t);
```

修正後:

```
extern void mnlsdemo_disp    (struct wkindarry *, nl_catd);
```

**mnlsdemo\_getdspwidth** 関数は、EUC のビット・パターンを使用していて修正できないため、削除します。

修正前:

```
extern int mnlsdemo_getdspwidth    (wchar_t *,
                                     eucwidth_t *);
```

修正後:

```
#ifdef MNLS_UNDELETE
extern int mnlsdemo_getdspwidth    (wchar_t *,
                                     eucwidth_t *);
#endif MNLS_UNDELETE
```

- **mnlsdemo\_main.c**

**getwidth.h** の展開を次のようにコメントにします。

```
/*
#include <getwidth.h>
*/
```

**setcat** 関数、**getwidth** 関数は、削除して **catopen** 関数に置き換えます。また、**main** 関数の最後で **catclose** 関数を呼び出します。

修正前:

```
setcat(MSGFILENAME);
getwidth(&widthset);
```

修正後:

```
nl_catd catd;
.
.
.
/*
setcat(MSGFILENAME);
getwidth(&widthset);
*/
catd = catopen(MSGFILENAME, NL_CAT_LOCALE);

.
.
.
catclose(catd);
```

**gettext 関数を catgets 関数に書き換えます。**

**修正前:**

```
fprintf(stderr, "%s:%s\n", gettext(MSGID_ERROOPEN, MSG_ERROOPEN),
```

**修正後:**

```
fprintf(stderr, "%s:%s\n", catgets(catd, 1,
                                MSGID_ERROOPEN, MSG_ERROOPEN),
```

**wscpy 関数を wcscpy 関数に書き換えます。**

**修正前:**

```
wscpy(wbuf.wkind_buf[kind][col], wstr_buf);
```

**修正後:**

```
wcscpy(wbuf.wkind_buf[kind][col], wstr_buf);
```

- **mnlsdemo\_getword.c**

**wslen 関数, wstok 関数, wscpy 関数を wcslen 関数, wcstok 関数, wcscpy 関数に書き換えます。**

プログラムの移行例  
B.3 XPG4 国際化インタフェースへの書き換えによる移行例

修正前:

```
if(wln_buf[wslen(wln_buf) -1] == L'\n') {  
    wln_buf[wslen(wln_buf) -1] = L'\0';  
    wptr = wstok(wln_buf, wtok);  
    wptr = wstok((wchar_t *)NULL, wtok);  
    wscopy(wstr_buf, wptr);
```

修正後:

```
if(wln_buf[wcslen(wln_buf) -1] == L'\n') {  
    wln_buf[wcslen(wln_buf) -1] = L'\0';  
    wptr = wcstok(wln_buf, wtok);  
    wptr = wcstok((wchar_t *)NULL, wtok);  
    wcscopy(wstr_buf, wptr);
```

- mnlsdemo\_class.c

ヘッダ・ファイルを XPG4 に対応したものに書き換えます。

修正前:

```
#include <wctype.h>  
#include <wdec.h>
```

修正後:

```
#include <wchar.h>
```

wslen 関数, isphonogram 関数, isideogram 関数, isnumber 関数を XPG4 国際化インタフェースに書き換えます。

修正前:

```
if(wslen(wstr_buf) == 0) {  
    for(len=0; len < wslen(wstr_buf); len++){  
        flag &= isphonogram(wstr_buf[len]);  
        flag &= isideogram(wstr_buf[len]);  
        || isnumber(wstr_buf[len]);
```

修正後:

```
if(wcslen(wstr_buf) == 0) {  
    for(len=0; len < wcslen(wstr_buf); len++);  
    flag &= iswctype(wstr_buf[len], wctype("phonogram"));  
    flag &= iswctype(wstr_buf[len], wctype("ideogram"));  
    || iswctype(wstr_buf[len], wctype("number"));
```

文字クラス "phonogram", "ideogram", "number" が ja\_JP.eucJP ロケールに存在していることが前提です。

- mnlsdemo\_disp.c

メッセージ・カタログの識別子を main 関数から引き継ぐため, mnlsdemo\_disp 関数の引数を変更します。

修正前:

```
void mnlsdemo_disp( struct wkindarry *wbuf, ecwidth_t *widthset );
```

修正後:

```
void mnlsdemo_disp( struct wkindarry *wbuf, nl_catd catd );
```

ワイド文字のビット・パターンにより文字列の表示幅を処理していますので, 次のように wcswidth 関数に置き換えます。

修正前:

```
dsp_size =  
    mnlsdemo_getdspwidth(wbuf->wkind_buf[kind]col,  
                          widthset );
```

修正後:

```
dsp_size =  
    wcswidth(wbuf->wkind_buf[kind]col,  
             wcslen(wbuf->wkind_buf[kind]col));
```

wscat 関数, wslen 関数, gettxt 関数を wscat 関数, wcslen 関数, catgets 関数に書き換えます。

プログラムの移行例  
B.3 XPG4 国際化インタフェースへの書き換えによる移行例

修正前:

```
wscat(wln_buf, wbuf->wkind_buf[kind][col]);
wscat(wln_buf, L" ");
wscat(wln_buf, L"\n");
wslen(wln_buf);
printf("%s\n", gettxt(MSGID_PHONO, MSG_PHONO));
printf("%s\n", gettxt(MSGID_IDEO, MSG_IDEO));
printf("%s\n", gettxt(MSGID_PUNCT, MSG_PUNCT));
printf("%s\n", gettxt(MSGID_LOWER, MSG_LOWER));
printf("%s\n", gettxt(MSGID_UPPER, MSG_UPPER));
printf("%s\n", gettxt(MSGID_NUMERIC, MSG_NUMERIC));
printf("%s\n", gettxt(MSGID_OTHERS, MSG_OTHERS));
```

修正後:

```
wcscat(wln_buf, wbuf->wkind_buf[kind][col]);
wcscat(wln_buf, L" ");
wcscat(wln_buf, L"\n");
wcslen(wln_buf);
printf("%s\n", catgets(catd, 1, MSGID_PHONO, MSG_PHONO));
printf("%s\n", catgets(catd, 1, MSGID_IDEO, MSG_IDEO));
printf("%s\n", catgets(catd, 1, MSGID_PUNCT, MSG_PUNCT));
printf("%s\n", catgets(catd, 1, MSGID_LOWER, MSG_LOWER));
printf("%s\n", catgets(catd, 1, MSGID_UPPER, MSG_UPPER));
printf("%s\n", catgets(catd, 1, MSGID_NUMERIC, MSG_NUMERIC));
printf("%s\n", catgets(catd, 1, MSGID_OTHERS, MSG_OTHERS));
```

書式付き出力関数でのワイド文字制御の変換仕様 (%ws %wc) を修正します。

修正前:

```
printf("%ws", wln_buf);
```

修正後:

```
printf("%S", wln_buf);
```

mnlsdemo\_getdspwidth 関数が不要になりますので、関数全体をコメントにします。

修正後:

```
#ifdef MNLS_UNDELETE
int mnlsdemo_getdspwidth(wchar_t *wkind_buf, eucwidth_t *widthset)
{
    .
    .
    この間には、関数の本体の記述があります。
    .
    .
}
#endif MNLS_UNDELETE
```

### B.3.4 メッセージ・ファイルの変換

System V MNLS 用に作成されたメッセージ・ソース・ファイルをメッセージ・ファイル移行支援ツールを用いて、XPG4 メッセージ・ソース・ファイルに変換します。次のように入力すると変換された XPG4 メッセージ・ソース・ファイル `mnlsdemo_msg.msg` がホーム・ディレクトリに生成されます。

```
% cp /usr/i18n/examples/mnls_xpg4/MNLS/mnlsdemo_msg.txt $HOME/mnlsdemo_msg.org
% /usr/i18n/bin/svr4/msgtrn $HOME/mnlsdemo_msg.org mnlsdemo_msg.msg
```

### B.3.5 プログラムのコンパイル

これまでの修正の結果が、`/usr/i18n/examples/mnls_xpg4/XPG4` に格納されています。これをコンパイルするには、次のようにしてホーム・ディレクトリにコピーしてから行って下さい。

```
% cp -r /usr/i18n/examples/mnls_xpg4/XPG4 $HOME
% cd $HOME/XPG4
% make all
% make messages
```

これにより、プログラムの実行モジュールとメッセージ・カタログが生成されます。

### B.3.6 プログラムの実行

生成された実行モジュールを実行するには、次の環境変数の設定が必要です。

- NLSPATH

メッセージ・カタログのサーチ・パスを設定します。サンプルのメッセージ・カタログは、プログラムと同じディレクトリ (\$HOME/XPG4) に生成されていますから、次のように設定します。

```
% setenv NLSPATH $HOME/XPG4/%N:$NLSPATH
```

- LANG

サンプルのメッセージやデータ・ファイルは ja\_JP.eucJP ロケールを使用しているので、次のように設定します。

```
% setenv LANG ja_JP.eucJP
```

- CSWIDTH

マイグレーション・キットの getwidth 関数が使用します。ja\_JP.eucJP ロケールでは、次のように設定します。

```
setenv CSWIDTH 2:2,1:1,2:2
```

以上の設定で実行モジュールを実行します。



C

---

リファレンス・ページ

---

## getwidth

ja\_JP.eucJP ロケールの補助コードセット情報を読み込みます。

---

### 形式

```
#include <sys/euc.h>
#include <getwidth.h>

void getwidth(eucwidth_t *ptr);
```

---

### 機能説明

環境変数 CSWIDTH から日本語 EUC の補助コードセット情報を読み込み、構造体 eucwidth\_t に設定します。

環境変数 CSWIDTH の形式

CSWIDTH= *n1* [:*s1*], *n2* [:*s2*], *n3* [:*s3*]

<i>n1</i>	補助コードセット 1 に対する EUC のバイト長
<i>s1</i>	補助コードセット 1 に対する 1 文字の表示幅 (省略時は <i>n1</i> の値が設定されます)
<i>n2</i>	補助コードセット 2 に対する EUC のバイト長 (SS2 は除きます)
<i>s2</i>	補助コードセット 2 に対する 1 文字の表示幅 (省略時は <i>n2</i> の値が設定されます)
<i>n3</i>	補助コードセット 3 に対する EUC のバイト長 (SS3 は除きます)
<i>s3</i>	補助コードセット 3 に対する 1 文字の表示幅 (省略時は <i>n3</i> の値が設定されます)

デフォルト値 (環境変数 CSWIDTH が設定されていない場合)

CSWIDTH=1,0,0

## 構造体の形式

```
typedef struct {
    short int _eucw1, _eucw2, _eucw3;
    short int _scrw1, _scrw2, _scrw3;
    short int _pcw;
    char _multibyte;
} eucwidth_t;
```

<code>_eucw1</code>	補助コードセット 1 に対する EUC のバイト長
<code>_eucw2</code>	補助コードセット 2 に対する EUC のバイト長 (SS2 は除きます)
<code>_eucw3</code>	補助コードセット 3 に対する EUC のバイト長 (SS3 は除きます)
<code>_scrw1</code>	補助コードセット 1 に対する 1 文字の表示幅
<code>_scrw2</code>	補助コードセット 2 に対する 1 文字の表示幅
<code>_scrw3</code>	補助コードセット 3 に対する 1 文字の表示幅
<code>_pcw</code>	プロセスコード幅 (常に 4 が設定されます)
<code>_multibyte</code>	EUC の最大バイト長 (SS2, SS3 を含みます) を設定します。

環境変数 CSWIDTH が設定されていない場合は次の値を設定します。

```
_eucw1=1, _eucw2=0, _eucw3=0;
_scrw1=1, _scrw2=0, _scrw3=0;
_pcw=4;
_multibyte=1;
```

`ptr` が NULL ポインタの時は、処理は行われません。

---

## 注意事項

- `ja_JP.eucJP` 以外のロケールはサポートしていません。
- ワイド文字のサイズは 32 ビットです。

---

## CODESETNO

日本語 EUC コードセットのコードセット番号に対応する値を返します。

---

### 形式

```
#include <sys/euc.h>

int CODESETNO(wc);
wchar_t wc;
```

---

### 機能説明

指定されたワイド文字を検査し、日本語 EUC コードセットのコードセット番号に対応して、0 ~ 3 の値を返します。

---

### 戻り値

0	コードセット 0 (ASCII)
1	コードセット 1
2	コードセット 2
3	コードセット 3

---

## msgtrn

メッセージ・ファイル移行支援ツール  
MNLS メッセージ・ソース・ファイルを XPG4 へ変換します。

---

### 形式

```
/usr/i18n/bin/svr4/msgtrn  mnls_msg [ xpg4_msg ]
```

---

### パラメータ

mnls\_msg  
System V MNLS メッセージ・ソース・ファイル

xpg4\_msg  
XPG4 メッセージ・ソース・ファイル  
省略時は標準出力へ出力します。

---

### 機能説明

System V MNLS メッセージ・ソース・ファイル (テキスト形式) を XPG4 メッセージ・ソース・ファイル (テキスト形式) へ変換します。

System V MNLS メッセージ・ソース・ファイルは次のように処理されます。

msgtrn

#### System V MNLS メッセージ・ソース・ファイル

メッセージ文字列 1  
メッセージ文字列 2  
メッセージ文字列 3

・  
・  
・

#### XP44 メッセージ・ソース・ファイル

\$set 1 1  
1 メッセージ文字列 1  
2 メッセージ文字列 2  
3 メッセージ文字列 3

2

・  
・  
・

- 1 ファイルの先頭には、メッセージ・セット番号が付加されます。msgtrn は、常に 1 を設定します。したがって、catgets 関数の引数のメッセージ・セット番号には、1 を指定して下さい。
- 2 各行のメッセージ文字列の先頭にメッセージの連番および空白文字が付加されます。

---

## 関連項目

コマンド : gencat , mkmsgs

関数 : catopen , catgets , catclose , gettxt

---

## chkmnls

プログラム移行支援ツール  
プログラム中の MNLS インタフェースを検索します。

---

### 形式

```
/usr/i18n/bin/svr4/chkmnls [ -m ] sourcefile [...]
```

---

### パラメータ

sourcefile  
MNLS アプリケーション・プログラムのソースファイルを指定します。

---

### オプション

-m           マーキング情報を出力します (省略時は、統計情報を標準出力に出力します)。

---

### 機能説明

System V MNLS の関数名やヘッダ・ファイル名の検索を行い、MNLS アプリケーションを XPG4 国際化インタフェースを使用して Tru64 UNIX に移行するための次のような情報を出力します。

- 統計情報 (オプション指定なしの場合)  
使用されている System V MNLS の関数、ヘッダ・ファイルの統計情報です。
- マーキング情報 (-m オプション指定時)

プログラムのソースファイルの修正の必要な箇所の直前にコメントが付加されたファイルが自動生成されます。

#### 統計情報

各プログラム・ソース・ファイルの System V MNLS の機能の使用件数および、移行作業の難易度の情報を標準出力に出力します。

出力される統計情報の意味は次のとおりです。

Input-file	プログラム・ソース・ファイル名
Target	検索で検出した System V MNLS の関数名またはヘッダ・ファイル名
Migration-Level	移行作業の難易度
Level	Migration-Level と同じ意味
Count	検出された件数
Grand Total	処理した全てのプログラム・ソース・ファイルの総合計

#### マーキング情報

プログラムのソースファイルの修正の必要な箇所の直前にコメントが付加されたファイルが自動生成されます。

生成されるファイル名は、sourcefile.chk となります。指定されたプログラムのソース・ファイルが test.c の場合、生成されるファイルは test.c.chk となります。

出力されるマーキング情報 (コメント) は次の形式です。

```
/* CHKMNLS*:xxxxx...:yyyyy...:(n) */
```

xxxxx...	検出された System V MNLS の関数名または、ヘッダ・ファイル名
yyyyy...	上記 xxxxx... に対する、移行に必要な簡単なコメント
n	移行作業の難易度



## 移行作業の難易度

(E)	Easy	比較的容易な修正で済みます。関数名やヘッダ・ファイル名の変更などがこれに分類されます。
(M)	Medium	(E) よりも、若干の修正が必要となります。 gettxt 関数などがこれに分類されます。
(D)	Difficult	ロジックの変更を必要とします。 XPG4 にはない機能はこれに分類されます。

## データ・ベース・ファイル

検索用データが格納されています。

ファイルの仕様は次のとおりです。

ファイル名:

/usr/i18n/lib/svr4/chkmnls.inf

形式:

xxxxx...:yyyyy...:n[LF]

xxxxx...	検索文字列
yyyyy...	上記 xxxxx... に対する、移行に必要な簡単なコメント
n	移行作業の難易度
[LF]	改行文字

先頭が#の行は無視されます。

---

制限事項

ソース・ファイル内のすべての文字列が検索の対象となります。したがって、検索対象のマクロ名や関数名と同じ名前の変数やコメント中の文字列に対して統計情報やマーキング情報が出力されることがあります。



Tru64 UNIX  
System V MNLS から Tru64 UNIX への国際化機能移行ガイド

---

1999 年 10 月 発行

コンパックコンピュータ株式会社

〒140-8641 東京都品川区東品川 2-2-24 天王洲セントラルタワー

電話 (03)5463-6600 (大代表)

---

AA-RK3WA-TE

