Guidelines for the Development of Localization Packages

UNIX International
Internationalization Work Group
Revision: 1.9 (Date: 93/04/16)

Published by:

UNIX International
Waterview Corporate Center
20 Waterview Boulevard
Parsippany, NJ 07054

for further information, contact:
Vice President of Marketing

Phone: (1) 201-263-8400
Fax: (1) 201-263-8401

International Offices:

UNIX International
Asian/Pacific Office
Shinei Bldg. 1F
Kameido
Koto-ku, Tokyo 136
Japan

Phone: (81) 3-3636-1122
Fax: (81) 3-3636-1121

UNIX International
European Office
25, Avenue de Beaulieu
1160 Brussels
Belgium

Phone: (32) 2-672-3700
Fax: (32) 2-672-4415

UNIX International
Pacific Basin Office
Cintech II
75 Science Park Drive
Singapore Science Park
Singapore 0511
Singapore

Phone: (65) 776-0313
Fax: (65) 776-0421

NOTICE:

UNIX International is making this documentation available as a reference point for the industry. While UNIX International believes that these information in this release of this document is well defined, minor changes may be made prior to products supporting these guidelines being made available from UNIX System Laboratories or UNIX International members.

Trademarks

UNIX® is a registered trademark of UNIX System Laboratories in the United States and other countries. X/Open™ is a trademark of the X/Open Company Limited.
AT&T® is a registered trademark of American Telephone & Telegraph in the United States and other countries.
OPEN LOOK™ is a trademark of UNIX System Laboratories.
PostScript™ is a trademark of Adobe Systems, Inc.
XENIX® is a registered trademark of Microsoft Corporation.
SunOS™ is a trademark of Sun Microsystems.
The X Window System™ is a trademark of MIT.

Preface

The *Guidelines for the Development of Localization Packages* document describes the localization features and functions which should be provided and considered as products to allow UNIX System V to operate using local language and customs. It also provides the developer with guidelines to develop a *Localization Package* for an individual region.

The internationalization features of UNIX System V provide a base system that is free of any language or cultural dependencies. For UNIX System V to be used in an international environment, Localization Packages must be provided. A Localization Package is designed to support a non-U.S. market. It should provide country- or region- or culture- specific features built on top of the internationalization capabilities of UNIX System V. Features such as native language system messages, character classification and conversion tables, and terminfo databases for local environments may be provided.

It is important that the Localization Package be developed and delivered in a timely manner, along with the base system.

Assumptions about the Reader

This document assumes the reader is working on the development of a Localization Package and is either:

- A UNIX SVR4 (or later version) system developer, or
- A system developer for a USL Regional office.

This document also assumes the reader has a basic knowledge of the following:

- ANSI-C (X3.159-1989) / ISO-C (ISO/IEC 9899:1990) Localization support,
- POSIX locale model, and
- Internationalization features of SVR4 Multi-National Language Supplement (MNLS).

Related Documents

A list of related documents is shown below:

- UNIX System V Release 4 Multi-National Language Supplement Product Overview
- UNIX System V Release 4 European Language Supplement Product Overview
- UNIX System V Release 4 Programmer's Reference Manual
- UNIX System V Release 4 System Administrator's Reference Manual
- UNIX System V Release 4 User's Reference Manual
- International Standard ISO 636:1988 - Code for the representation of names of languages
- International Standard ISO 3166:1988 - Codes for the representation of names of countries

- International Standard ISO 4217:1990 - Codes for the representation of currencies and funds
- International Standard ISO/IEC 9899:1990 - Programming languages - C
- International Standard ISO/IEC 9945-1:1990 - Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language]
- IEEE Standard Std 1003.1:1990 - Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language]
- American National Standard ANSI X3.159-1989 for Information System - Programming Language - C
- X/Open Portability Guide Issue 3 - XSI Commands and Utilities, Volume 1.
- X/Open Portability Guide Issue 3 - XSI Supplementary Definitions, Volume 3.
- System V Interface Definition Third Edition Volumes I,II,III,V.

1. Introduction

This chapter describes the internationalization and localization features that are contained in the document *UNIX System V Release 4* (SVR4), *UNIX System V Release 4 Multi-National Language Supplement 4* (SVR4 MNLS) and the *System V Release 4.0 European Language Supplement Product Overview and Installation Guide*. Additionally, all of the internationalization features contained in SVR4 and MNLS 4.0 are provided in UNIX System V Release 4.1 Enhanced Security (SVR4.1ES). The base internationalized system (SVR 4 + MNLS 4.0 or SVR4.1ES) will be referred to in this document as "System V". Any differences between these will be noted in the document.

1.1 Internationalization

Internationalization refers to the capabilities of a system that:

- allow the system to support any native language, cultural information or codeset. The combination of these features form an environment called a "*locale*".
- promote the *development* and *use* of applications that function in different locales.

System V provides the following important internationalization features in the base system:

- Support of full 8-bit code-sets and multibyte code-sets.
- Support of locale-specific date and time formats.
- ANSI/ISO C Internationalization and Localization functions.
- Libraries to support internationalization enhancements, and multibyte/wide character processing.
- Message Handling Facilities/Tools.
- STREAMS-based TTY drivers and a line discipline module for handling EUC encoding character I/O.
- Keyboard mapping.
- Code-set conversion utility.
- UNIX commands which support the use of *Extended UNIX Code* (EUC) set, are sensitive to different native language or culture-specific environments and which can display native language messages to a user.

1.2 Localization

Localization is the process of providing native language and cultural-specific information to a system.

System V does not have language and/or custom-dependent information. Language or culture- dependent information is provided at runtime by the locale. A *Localization Package* contains information to support one or more *locale*s.

Localization should be based on the System V's internationalization features. The localization information/features in a *Localization Package* may contain:

- Locale-specific information for ANSI/ISO C and wide character processing functions,
- Native language system message catalogs,
- Language-specific character classification functions,
- Locale-specific terminal/printer information,
- Language-specific Input Method / Keyboard Mapping,
- Locale-specific code conversion information.

2. Building a Locale

2.1 Overview

A *locale* contains a set of information that defines the various culture-specific conventions, such as date and time, collation information, character classification, and native language messages.

This chapter outlines the steps that must be taken to build the set of locale-specific information required to localize a System V Operating System.

2.1.1 Locale Categories

ANSI/ISO C localization functions for the C library are provided in System V. These functions allow the development of language- and culture- independent software. These comply with the ANSI C standard.

Using the ANSI/ISO C localization functions, applications can be used in any native language or culture-specific environment as if in an English environment, with no source code changes and no recompilation.

A *locale* can be described by the following categories of information:

LC_CTYPE:	Character Classification and Conversion
LC_NUMERIC:	Numeric Representation
LC_TIME:	Date and Time Format
LC_COLLATE:	Collating Sequence
LC_MONETARY:	Monetary Format
LC_MESSAGES:	Message databases information

For POSIX 1003.1- and XPG3- conformant systems, environment variables are defined that correspond to some of the named categories above and that have the same spelling. Two additional environment variables, *LC_ALL*, and *LANG* are also defined. LC_ALL refers to the name of the locale to be used to override any values for locale categories specified by the setting of LANG or any environment variables beginning with "LC_".

Since the LC_ALL environment variable was introduced and specified by the IEEE Std 1003.1-1990 (ISO/IEC 9945-1:1990) for the first time, System V 4.1ES and earlier versions did not support this environment variable. Future versions of System V will provide this feature.

See the next section for more information.

The *setlocale()* function sets, changes or queries the locale of a process according to the values of the category and locale in:

char *setlocale(int *category*, **const char** **locale*);

Refer to

- Section "Internationalization" of *SVR4 ANSI C Transition Guide*,
- Section 4.4, "Localization <locale.h>" of ANSI X3.159-1989,
- Section 7.4, "Localization <locale.h>" of ISO/IEC 9899:1990,
- Section 4.2, "PROGRAM LOCALISATION", of the *X/Open Portability Guide - Volume 3 XSI Supplementary Definitions*, and
- Sections 2.6, "Environment Description", E.1.3, "Locale Definitions", ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990)

for more information.

2.1.2 Environment Variables for Locales

Any application that processes native language-specific information must be able to execute in any locale without being recompiled. This information includes date and time formats, or monetary symbols, and collation that differs among languages.

In an internationalized system, the environment variables are as follows:

- LANG - the name of the locale to use for locale categories when either LC_ALL or the corresponding environment variable (beginning with "LC_") do not specify a locale.
- LC_COLLATE - the name of the locale for collation information.
- LC_CTYPE - the name of the locale for character classification.
- LC_MONETARY - the name of the locale containing monetary-related numeric editing information.
- LC_NUMERIC - the name of the locale containing numeric editing (i.e., radix character) information.
- LC_TIME - the name of the locale for date/time formatting information.

Note: Most of the above descriptions were excerpted from ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990).

LC_MESSAGES, the name of the locale containing the native language message databases, is defined in the the P1003.2 draft.

Refer to

- environ(4) in the *SVR4 System Administrator's Reference Manual*,
- Section 8.1.2, titled "Extensions to *setlocale()* Function", of the *ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990)*, and
- Section 4.2, titled "PROGRAM LOCALISATION", of the *X/Open Portability Guide - Volume 3 XSI Supplementary Definitions*

for more information.

2.2 Locale Database Files

The locale databases contain the required localization data for languages and code-sets.

The localization functions retrieve the localized information from a set of database files or directories:

File	**Generated By**
/usr/lib/locale/*locale*/LC_CTYPE	chrtbl, wchrtbl
/usr/lib/locale/*locale*/LC_NUMERIC	chrtbl, wchrtbl
/usr/lib/locale/*locale*/LC_TIME	(editor)
/usr/lib/locale/*locale*/LC_COLLATE	colltbl
/usr/lib/locale/*locale*/LC_MONETARY	montbl
/usr/lib/locale/*locale*/LC_MESSAGES/*message database*	gencat, mkmsgs
/usr/lib/locale/*locale*/LC_MESSAGES/Xopen_info	mkmsgs

A number of tools are provided to help create these files:

chrtbl(1M) generate a character classification and conversion table and numeric classification information

wchrtbl(1M) generate a character classification and conversion table and numeric classification information for ASCII and supplementary code sets

colltbl(1M) generate a collation sequence table for a language

montbl(1M) generate a monetary formatting table

gencat(1) generate a formatted message catalog for use by catgets(3C)

mkmsgs(1) generate a message file for use by both gettxt(1) and gettxt(3C)

2.3 Locale Naming Conventions

There is no standard for naming locales. X/Open specifies the following convention:

language[*_territory*][*.code-set*]

The language abbreviation is a two-lowercase-letter abbreviation for the language name. It comes from ISO 639.

When country abbreviations are used for the *territory*, the standard ISO 2-letter country abbreviations are recommended. It comes from ISO 3166.

The Appendix contains a tables of locale names that are used by some companies. These locales are not provided as part of System V. To avoid conflict, no new locales should use the same names.

Refer to

- *ISO 639 - Code for the representation of names of languages*,
- *ISO 3166 - Code for the representation of names of countries*,
- Section 4.2, titled "PROGRAM LOCALISATION", of the *X/Open Portability Guide - Volume 3 XSI Supplementary Definitions*,
- Appendix A of this document, "Locale Names used by ELS and Other Products",
- Appendix B of this document, "Two-letter language symbols", and
- Appendix C of this document, "Two-letter country code"

for more information.

2.4 Building the Character Classification and Conversion Table

The following steps are required to create the character classification and conversion table LC_CTYPE.

- Create a classification and conversion table according to the syntax defined in *chrtbl*(1M) for single-byte languages, or *wchrtbl*(1M) for multibyte languages.
- Run this newly formed table through the *chrtbl*(1M) or *wchrtbl*(1M) command, as appropriate.
- Move the newly created *file* to the file /usr/lib/locale/*locale*/LC_CTYPE, where *locale* is the locale in question.

Refer to

- Chapter 3, titled "Extended UNIX Code (EUC)", and Chapter 7, titled "Building a Locale", of the *SVR4 MNLS Product Overview*, and
- Appendix D of this document, titled "LC_CTYPE Example"

for more information.

2.5 Building The Numeric Table

The table containing the locale- dependent radix character and thousands separator is built by the *chrtbl*(1M) or *wchrtbl*(1M) command.

The steps required to build a table are as follows:

- Add the lines to the character classification description and conversion table to describe the numeric information, according to the syntax defined in the *chrtbl*(1M) or *wchrtbl*(1M) manual page.
- Run this newly formed table through the *chrtbl*(1M) or *wchrtbl*(1M) command.
- Move the newly created *file* to the the file /usr/lib/locale/*locale*/LC_NUMERIC, where *locale* is the locale in question.

Refer to

- *chrtbl*(1M) in *SVR4 System Administrator's Reference Manual*,
- Chapter 7, titled "Building a Locale" and *wchrtbl*(1M) in the *SVR4 MNLS Product Overview*
- Appendix D of this document, titled "LC_CTYPE Example"

for more information.

2.6 Building the Header File for Native Language Character Classifications and Conversions

Programmers should be discouraged from using their native language character classifications and conversion functions since these functions:

- are not part of any standards or SVID,
- are locale-specific functions which cannot be used in internationalized application programs.
- may be replaced by the ISO-C Multibyte Support Extension (MSE) in the future,

There are several header files that contain information used by character classification and conversion routines. These are listed below:

- /usr/include/ctype.h
- /usr/include/wctype.h
- /usr/include/*ctype.h

Caution: The header file /usr/include/xctype.h was originally designed to specify the definition of native language character and symbol classification and conversion routines. To avoid conflict among different locales that might share character classification functions with identical names, an application should include its locale-specific header, for example, /usr/include/*jctype.h* for a Japanese locale, /usr/include/*kctype.h* for a Korean locale, etcetera. This localized header should *not* be included in /usr/include/wctype.h.

USL's localization packages implement such header files.

There are two functions supplied by System V which allow a *Localization Package* developer to define the native language character classifications and conversion rules. The rules themselves are coded into the *character class table* that is created by *chrtbl*(1M) and/or *wchrtbl*(1M) utility. These two functions get their information from the *character class table*.

The functions are *_iswctype*() and *_trwctype*(), and their format is as follows:

_iswctype(c,*class*)
_trwctype(c,*class*)

The _iswctype() function returns non-zero if *c* is a member of the set of characters defined by *class*.

The _trwctype() function returns a corresponding converted character if *c* is a member of the set of characters defined by *class*

Refer to

- *chrtbl*(1M) in *SVR4 System Administrator's Reference Manual*,
- Chapter 3, titled "Extended UNIX Code (EUC)", and Chapter 7, titled "Building a Locale" in the *SVR4 MNLS Product Overview*,
- *wchrtbl*(1M) in *SVR4 MNLS Product Overview*, and
- Appendix E of this document, titled "xctype.h Example"

for more information.

2.7 Building The Date and Time Table

Date and time formats vary between different languages and countries. The C library function *strftime*(3C) allows the application developer to generate locale-dependent date and time information without embedding that information within their application. The format for the date and time table is the same as that used by the function *cftime*(3C). However, *cftime*(3C) is obsolete; all new programs should use *strftime*(3C).

The steps required to build the LC_TIME locale are as follows:

- Create a date and time table according to the format described on the *strftime*(4) manual page of the *SVR4 Programmer's Reference Manual*.
- Move the newly created *file* to the file /usr/lib/locale/*locale*/LC_TIME, where *locale* is the locale in question.

Refer to

- *strftime(3C)*, *strftime*(4) in the *SVR4 Programmer's Reference Manual*, and
- Appendix F of this document, titled "LC_TIME Example"

for more information.

2.8 Building The Collation Table

The collating sequence used in a particular language may differ from the US ASCII sequencing provided in the default UNIX System V environment. The traditional sequencing of US ASCII is related to the numerical representation of the characters. Other languages not only require different relative orderings of the characters within the US ASCII code-set, but will often need to support additional code-sets. For example, to support the majority of European languages the ISO 8859/1 code-set is needed.

To support different collating sequences, different transformation tables are required. The tables are used together with the *strxfrm*() library function to transform a given string to a second string that can be correctly ordered with other transformed strings by the existing *memcmp*(), *strcmp*(), and *strncmp*() functions. These functions can be found in *string*(3C).

The *colltbl*(1M) command creates the transformation table(s) based on different user specifications. The steps required for the use of the *colltbl*(1M) command are as follows:

- Create a collation table according to the syntax as defined in *colltbl*(1M).
- Run this newly formed table through the *colltbl*(1M) command.
- Move the newly created *file* to the file /usr/lib/locale/*locale*/LC_COLLATE, where *locale* is the locale in question.

Note: *colltbl* does not support multibyte characters.

Refer to

- *colltbl*(1M) in the *SVR4 System Administrator's Reference Manual*, and
- Appendix G of this document, titled "LC_COLLATE Example".

for more information.

2.9 Building The Monetary Table

The table containing the locale-dependent monetary formatting conventions is built by the *montbl*(1M) command.

The steps required to use the *montbl*(1M) command are:

- Create a monetary formatting description file for the new locale according to the syntax as defined in *montbl*(1M),
- Run this newly formed table through the *montbl*(1M) command, and
- Install the newly created *file* to the the file /usr/lib/locale/*locale*/LC_MONETARY, where *locale* is the locale in question.

Refer to

- *montbl*(1M) in the *System Administrator's Reference Manual*, and
- Appendix H of this document, titled "LC_MONETARY Example"

for more information.

2.10 Building The X/Open Language Information Table

Most of the XPG3 localization functions use the same locale database files as the ANSI-C / ISO-C localization functions. Some information, however, is specified separately in XPG3. This includes the items T_FMT, D_FMT, D_T_FMT, YESSTR, and NOSTR to be retrieved by the XPG3 function *nl_langinfo*(3C).

The specific XPG3 locale file containing this data is generated for each locale supported and installed in the file /usr/lib/locale/*locale*/LC_MESSAGES/Xopen_info.

This message catalog can be generated by mkmsgs(1M) (refer to chapter 3). The sequence of fields in the message catalog are:

<msgnumber>	Items	Description
1	**T_FMT**	Locale's default format for the time.
2	**D_FMT**	Locale's default format for the date.
3	**D_T_FMT**	Locale's default format for the date and time.
4	**YESSTR**	Locale's equivalent of 'yes'
5	**NOSTR**	Locale's equivalent of 'no'

Refer to the next chapter for the steps that are required to make a message catalog.

Refer to *langinfo*(5) in the *SVR4 System Administrator's Reference Manual* for more information

3. Native Language Message System

3.1 Overview

System V has internationalized many commands and utilities to provide native language message capabilities. *Localization Packages* should include the message files required to enable these internationalized commands and utilities to provide messages to users in the language of their designated locale.

This chapter describes the steps that are required to build native language message files.

3.1.1 The Commands/Utilities Supporting Native Language Messages

The following commands/utilities in System V are internationalized to provide native language messages:

accept	alpq	ar	at	atq
atrm	awk	banner	basename	batch
cancel	cat	cd	chgrp	chmod
chroot	cmp	col	comm	cp
cpio	cron	crontab	csplit	cut
date	dcopy	devnm	df	diff
dirname	disable	du	ed	editr
egrep	enable	ex	expr	fgrep
file	find	gencat	getopt	getopts
gettxt	iconv	join	jsh	kbdcomp
kbdload	kbdpipe	kbdset	kill	lfmt
ln	login	lp	lpadmin	lpfilter
lpforms	lpmove	lpsched	lpshut	lpstat
lpsystem	lpusers	ls	mail	mailx
mesg	mkdir	mv	mvdir	newgrp
news	nl	nohup	od	pack
passwd	paste	pcat	pfmt	pg
pr	ps	pwd	red	reject
rm	rmail	rmdir	rsh	sed
sh	shl	sleep	sort	split
stty	su	sum	tabs	tail
tar	tee	test	touch	tr
tty	ttymon	uname	uniq	unpack
vedit	vi	view	wait	wall
wc	who	write	yacc	

Refer to Appendix B, titled "Command List" of *SVR4 Product Overview* for more information.

There are two kinds of native language message files in the system:

- System V-specific Message Catalogs created by *mkmsgs*(1).
- X/Open Message Catalogs created by *gencat*(1).

The *Localization Package* for System V contains at least sixteen message catalogs and two special native language text files for each locale. The file names and paths will be specified in later sections.

Refer to

- Appendix B, titled "Command List" of the *SVR4 MNLS Product Overview*,
- The *SVR4 User's Reference Manual*,
- The *SVR4 System Administrator's Reference Manual*,
- The *SVR4 Programmer's Reference Manual*,
- The *SVR4 MNLS Product Overview*, and
- Chapter 5, titled "MESSAGE CATALOGUES", of the *X/Open Portability Guide Issue 3 - Volume 3 XSI Supplementary Definitions*

for more information.

3.1.2 Environment Variables

The location of the message catalogs is controlled by the environment variables NLSPATH, and LANG.

Refer to

- *environ*(4) in the *SVR4 System Administrator's Reference Manual*, and
- Chapter 8, titled "Language-Specific Services for the C Programming Language", in *SVR4 Programmer's Guide: POSIX Conformance*

for more information.

3.2 Native Language Message Catalogs

Commands and utilities supporting native language messages in System V use the message handling facility. This section provides the information for a *Localization*

Package developer to build the translated message catalogs.

3.3 Message Catalogs

The message catalogs used in SVR4.1ES are listed below:

Message Catalog(s)
uxar
uxaudit
uxawk
uxcore
uxcore.abi
uxcplu
uxdfm
uxdist
uxed.abi
uxemail
uxes
uxlibc
uxlp
uxmesg
uxmmm
uxnsu
uxroi
uxsmtp
uxsysadm
uxsyserr
uxue
uxue.abi

The English message source files for the C locale can be found under /usr/lib/locale/C/MSGFILES in the SVR4.1 ES source code product. In earlier releases, the messages are contained in the commands themselves.

All the messages in these catalogs should be translated into the desired native language and included in the catalogs as part of the *Localization Packages*. These translated message catalogs should be installed in the directory: /usr/lib/locale/*locale*/LC_MESSAGES where *locale* is a directory with the same name as the supporting locale.

These message catalogs are generated by the *mkmsgs*(1) command and can be retrieved by *gettxt*(3), or by *gettxt*(1), at the shell level.

Caution: These catalogs are maintained by USL. New messages may be added by USL to the end of the catalogs in later releases. Existing messages will not be modified or deleted. The prefix "ux" is reserved by USL for System V message catalog names.

Warning: Messages in the System V catalogs should not be changed. The default messages in the System V commands and utilities should not be changed. Any new messages added by an ISV should be placed in other catalogs.

3.3.1 Building a Native Language Message Catalog

System V provides two message handling facilities:

- System V-specific message handling facility.
- X/Open message handling facility (see the final section in this chapter).

These message handling facilities are used by system or application programmers.

Several functions and tools are provided in these two facilities:

MNLS message handling facility:

— Tools to create message catalogs:

  - A tool to extract and replace text strings: *exstr(1)*.
  - A tool to create message files for use by gettxt(see below): *mkmsgs*(1M).
  - A tool to search for a text string from a locale message database: *srchtxt*(1).

— A command to retrieve messages: *gettxt*(1).

— A function to retrieve messages: *gettxt*(3C).

The steps required to build the native language system messages are:

- Translate the default messages into the desired language. Default English messages can be found in the source code. In SVR4.1ES and later releases most of the message source files are found under /usr/lib/locale/C/MSGFILES. In earlier releases, the command *srchtxt* was to find the English messages embedded in the utilities themselves.

  The sequence of arguments in a message may need to change, for translation purposes. Reference *printf*(3C) in the *SVR4 Programmer's Reference Manual* for a description of positional arguments of the form "%*n*$", where *n* is the digit representing the argument position.

- Translate the default messages into the native language and store it in a text file.
- Generate the message catalogs by running the *mkmsgs*(1) command. Refer to *mkmsgs*(1) in the *SVR4 User's Reference Manual* for more information.
- Install the newly created message catalogs into the message directory of the supporting *locale*. The directory is /usr/lib/locale/*locale*/LC_MESSAGES where the *locale* is a directory that has the same name as the supporting *locale*. The names of runtime message files under the LC_MESSAGES directory should be the same as the names of message files (*catalog names*) used by commands or utilities.

Below are examples of the C locale message files in SVR4.1 ES.

Runtime File	Source File
LC_MESSAGES/uxaudit	/usr/src/cmd/messages/uxaudit/msgs
LC_MESSAGES/uxawk	/usr/src/cmd/messages/uxawk/msgs
LC_MESSAGES/uxcore	/usr/src/cmd/messages/uxcore/msgs
LC_MESSAGES/uxcore.abi	/usr/src/cmd/messages/uxcore.abi/msgs
LC_MESSAGES/uxdfm	/usr/src/cmd/messages/uxdfm/msgs
LC_MESSAGES/uxed.abi	/usr/src/cmd/messages/uxed.abi/msgs
LC_MESSAGES/uxes	/usr/src/cmd/messages/uxes/msgs
LC_MESSAGES/uxmesg	/usr/src/cmd/messages/uxmesg/msgs
LC_MESSAGES/uxnsu	/usr/src/cmd/messages/uxnsu/msgs
LC_MESSAGES/uxsysadm	/usr/src/cmd/messages/uxsysadm/msgs
LC_MESSAGES/uxue	/usr/src/cmd/messages/uxue/msgs
LC_MESSAGES/uxue.abi	/usr/src/cmd/messages/uxue.abi/msgs
LC_MESSAGES/Xopen_info	/usr/src/cmd/localedef/datetbl/time_C
LC_MESSAGES/uxar	/usr/src/cmd/sgs/catalogs/ar.str
LC_MESSAGES/uxcplu	/usr/src/cmd/sgs/catalogs/cplu.str
LC_MESSAGES/uxdist	/usr/src/cmd/dist/hdrs/dist.msgs
LC_MESSAGES/uxlibc	/usr/src/cmd/sgs/catalogs/libc.str
LC_MESSAGES/uxlp	/usr/src/cmd/lp/lib/oam/msg.source
LC_MESSAGES/uxmail	/usr/src/cmd/mail/pfmt.msgs
LC_MESSAGES/uxmmm	/usr/src/cmd/msgmgt/message/msglog_db
LC_MESSAGES/uxroi	/usr/src/cmd/roi/roi_msgs/roimsgs
LC_MESSAGES/uxsmtp	/usr/src/cmd/mail/smtp/src/pfmt.msgs
LC_MESSAGES/uxsyserr	/usr/src/lib/libc/port/gen/syserr.str

Refer to Chapter 6 ("Commands") in the *SVR4 MNLS Product Overview* for more information.

3.4 Special case for mailx(1)

In addition to the message catalogs, mailx(1) of System V uses native language help messages in two ordinary text files.

The source files for the mailx(1) help messages in the C *locale* (English messages) are:

```
/usr/src/cmd/mailx/help/mailx.help
/usr/src/cmd/mailx/help/mailx.help.
```

These messages are stored as help files in:

```
/usr/share/lib/mailx/C/mailx.help
/usr/share/lib/mailx/C/mailx.help.
```

The native language help message files for other locales
should be placed in:

/usr/share/lib/mailx/*locale*/mailx.help
/usr/share/lib/mailx/*locale*/mailx.help.

where *locale* corresponds to the name of the native language environment in which these help messages are to be displayed to the users.

These help message files are stored in ordinary text format.

Refer to

- *mailx*(1) in the *SVR4 MNLS Product Overview*

for more information.

3.5 Special case for X/Open Messaging

In the X/Open internationalized messaging system, message catalogs are created using the *gencat*(1) command. This command takes one or more source files containing the message text, and produces or modifies a message catalog. With the `-m` option, *gencat* builds a single catalog file compatible with the catalog format generated by the previous version of *gencat*. The retrieval routines detect the type of catalog they are using and act properly.

The location of message catalogs and the conventions for naming them are not defined by X/Open and are implementation dependent.

The *catopen*(3C) and *catclose*(3C) functions are used to open and close a message catalog file. Message catalogs are opened, ready for use, by calling the *catopen*() function, which locates the identified message catalog according to the search and naming rules defined in the NLSPATH environment variable. The default location for message catalogs is the current directory if NLSPATH is not set.

The *catgets*(3C) function is used to retrieve a translated message string from the message catalog opened by *catopen*().

4. Code-set Conversion

In an international environment several encoding schemes may need to be supported on the same system. Furthermore, the conversion from one code-set to another may also be required.

In System V, two facilities are provided for doing code-set conversion.

- STREAMS Input/Output Code-set Conversion
- File Codeset Conversion

This section describes these features and provides a description of the localization data that is required to support these capabilities.

4.1 Table Driven Conversion

The STREAMS based code-set conversion and the file code-set conversion facilities utilize a common underlying table driven approach. In this approach, conversion rules are provided in a file which describe how codes are to be transformed. The steps required to create a conversion table are as follows.

- Create a code-set conversion description file according to the syntax defined in kbdcomp(1M).
- Compile the code-set conversion table into a binary file using the kbdcomp(1M) command.
- Install the newly created file under the directory /usr/lib/kbd.
- Test the new conversion table with the iconv(1M) command or the kbd(7) STREAMS module.

See also limitations of table conversion in the later sub section.

4.1.1 Code-set conversion Table

A code-set conversion table provides a uni-directional mapping from one code-set to another. If code-set conversion is required in both directions, then conversion tables should be created for each direction.

Additionally, you may want to provide variants for each code-set conversion table to handle the cases where the source character code cannot be represented in the destination code-set. The USL SVR4.0 European Language Supplement (ELS) Product, for example, has 4 variants for each table.

1. "default" tables convert source code-set characters which cannot be represented in the target code-set to a default character specified in the tables.
2. "best fit" tables convert characters in the source code set to the closest approximation in the target code-set, but only use a one-to-one mapping. (for instance, 'a grave' n an 8-bit code-set may map to 'a' in a 7-bit code-set).
3. "best fit with expansion" tables convert characters in the source code-set to the closest matching character in the target code-set, possibly to a sequence of

characters in the target code set. character code. (for example, 'a umlaut' may map to 'ae')

4. "printer mode" tables may display characters by overstriking, if a non-destructive backspace is available. ('a umlaut' may be represented by 'a backspace "')

Refer to the ELS package for additional information.

4.2 STREAMS Input/Output Codeset Conversion

STREAMS based code-set conversion can be used to convert character codes passing through a stream. This feature makes it possible to connect terminals and/or printers to a system which have incompatible code-sets.

The STREAMS base conversion facility utilizes the kbd(7) streams module. This module is pushed onto a stream (usually a tty), and allows one to attach conversion tables to either the input and/or output direction of the stream.

Refer to strconf(1), strchg(1), kbdload(1) and kbdset(1) for more information.

Also, refer to chapter 7 for STREAMS configuration.

4.3 File Codeset Conversion

Files imported from another system may require the codes to be translated to another code-set. To accomplish this task, the iconv(1) command can be used.

The iconv command, utilizes the database file /usr/lib/iconv/iconv_data to identify the conversion table. This database file has 4 fields:

<fromcode> *<tocode>* *<table_name>* *<file>*

where

<fromcode> is the name of the original code-set.

<tocode> is the name of the destination code-set.

<table_name> is an internal name defined in the
conversion table. The recommended naming convention
for <table_name> is:

<fromcode>.<tocode>

<file> is the object file name of the code-set conversion table.

In addition to providing the conversion tables, the L10N package needs supply the iconv_data description lines. For each conversion table, one description line should be appended to the iconv_data table. This should be done by the L10N package at installed time.

Note: The above interfaces may change when XPG4 is finalized.

4.3.1 Limitations of Table Conversion

The table conversion facilities in System V are limited to simple byte-to-byte and string-to-string mappings. More complex encoding schemes or state-dependent encoding (i.e.

shift-in/shift-out protocols) can not be handled using this approach.

For more complex conversions, System V provides an algorithm- based conversion capability. For more information on this facility, refer to alp(7).

In addition to the above limitations, the following limits for a code-set conversion tables should be observed.

- A maximum length of 128 bytes for input strings.
- A maximum length of 256 bytes for output strings.
- The total amount of space consumed by a single table is limited to approximately 65,000 bytes.
- Object tables are machine-dependent in their byte order and structure size. When using remote devices across a network between heterogeneous machines, tables must be loaded on the machine where the module is actually pushed.

5. Input Methods

5.1 Simple Input Methods

The variety of input methods required in the European market is easily supported by System V. As described in the preceding chapter, keyboard mapping may be supported by use of the *iconv* utility or through use of the kbd STREAMS modules. *Dead key* and *compose key* mapping is easily described in the system.

5.1.1 Dead Keys

Some keyboards support characters that are used in conjunction with another key. Such characters are "*dead characters*", and the corresponding keys are called "*dead keys*". For example, a dead key may be used to produce an accented character, or a character with a circumflex. Examples of dead keys are:

acute(´)
grave(`)
circumflex(ˆ)

When any of these keys, designated as a dead key, is pressed, no symbol will appear on the screen until a trailing, valid character is pressed.

Note: Description taken from the System V Release 4.0 European Language Supplement Product Overview and Installation Guide, USL, July 1990.

5.1.2 Compose Keys

A *compose key* and a designated set of trailing characters, is used to allow a user to enter a character that does not have a corresponding key on the user's keyboard. For example, in the SVR4.0 European Language Supplement, compose key + "L" + "-" form the input code for the sterling currency symbol.

5.1.2.1 Keyboard Mapping

SVR4/386 provides the utilities *mapkey(1)* and *Imapchan(1)* to specify the character codes that are returned by a particular keyboard. The mapkey(1) command supports PC keyboard configuration, while the mapchan(1) utility configures the mapping of information input and output of *tty* devices. Dead key mapping is provided by *mapchan(1)*.

SVR4.0 European Language Supplement (ELS) provides keyboard mapping data files for several European keyboards and code-sets.

Compose key and dead key support can be further provided by the kbd STREAMS module described in chapter 4.

5.2 Complex Input Methods

The most difficult aspect of supporting Asian languages is providing support for their input methods.

The major characters used in dominant East Asian languages (in computer use) are multibyte characters, such as *Hanzi* in China and Taiwan, *Hangul* and *Hanja* in Korea, *Katanaka*, *Hiragana* and *Kanji* in Japan. Asian ideographic character sets such as Hanzi, Hanja and Kanji have several hundred thousand characters. Asian input methods through an ASCII-like keyboard therefore require extremely complex mechanisms and processes beyond *dead key*, *compose key* and *keyboard mapping* techniques.

The following are examples of Asian input method components.

— ASCII character to Asian alphabet (phonetic character) conversion

— Dictionary look up (mainly for ideographic characters)

— Dialog with the user to select the most appropriate (ideographic) character representation from among several candidates (homonyms).

The variety of input methods and the lack of standardized mechanisms and interfaces for such complex methods have led to proprietary implementations, third party products and/or public software products in the market.

Although System V itself as a base system does not include any complex input methods, it does provide several important capabilities for supporting complex input methods:

- STREAMS-based TTY interface
- Pseudo TTY (*pty*) interface
- Socket interface

The following is a list of feasible choices and steps to provide a complex input method as a part of vendor specific (localization) packages.

Option 1: Self implementation

Taking advantages of (some of) System V's capabilities, vendors can develop their own complex input method to meet their customers' needs.

Option 2: USLP's licensed product

Using the STREAMS-based TTY technology, UNIX Systems Laboratories Pacific (USLP) provides the following add-on packages of Asian character input systems as separately licensed products. Vendors may choose the USLP products and then customize where necessary.

- JI/O STREAMS-Based Japanese I/O Subsystem.

  A set of generic STREAMS modules and a pseudo device driver for implementing a dictionary looking up daemon process.

  This subsystem includes the following:

  — Kana-to-Kanji translation upon key entry from Japanese terminal,

  — Code conversion for output to Japanese terminal/printer,

— Japanese dictionary for Kana-to-Kanji translation,

— Dictionary maintenance utilities.

- VJE-β [1] STREAMS-Based Japanese Front-end Processor for Japanese character cell terminals.
- S-jip [2] STREAMS-Based Japanese Input Subsystem for Japanese character cell terminals.
- KI/O STREAMS-Based Korean I/O Subsystem.

For more details, contact UNIX System Laboratories Pacific, Ltd.

Option 3: X11R5 based

The X11 Release 5.0 includes a lot of contribution software for complex Asian input methods in the window environment in conjunction with the XIMP specification* which has been developed by UI members and approved and endorsed by the UI IWG. Vendors may choose and customize some of the following contribution software.

- alya Oki's Japanese input server
- Htt Sun's multi-lingual input server
- Kmf FujiXerox's Japanese input server
- sjx Sony's Japanese input server
- xfeoak Fujitsu's Japanese input server

Option 4: Third party product

In each country and region, there may be several companies selling complex input method products in the UNIX environment. Vendors may find good partners through appropriate channels. For details, contact your local subsidiaries and/or the regional UNIX community.

Option 5: Public software

There are some public softwares of Asian input method as listed below.

- Wnn A socket- (and pty-) based Japanese input method which was developed by the joint group of Kyoto University RIMS, Omron Corp., and Astec, Inc.

---

1. VJE-β is developed and copyrighted by VACS Corporation and ASCII Corporation.
2. S-jip is developed by SCR Corporation.

* Requirements for an X Window System Input Method Protocol, Version 1.0, Internationalization Work Group, UNIX International, Inc..

There are some variants of the Wnn (plus so-called *Jserver*), one of which was included in the X11 Release 4 as *Xwnmo* in the user contribution software.

For details, contact the Wnn Consortium in Japan.

- SKK — A software package which provides a Japanese input environment for *Nemacs*, a Japanese version of GNU Emacs. The SKK was developed by Masahiko Sato, et al. at RIEC of Tohoku University, Japan.

  For details, contact RIEC of Tohoku University, Japan.

- kinput2 — The latest version of *kinput* which is an X window application to input Japanese text, acting as a front-end for other applications. *kinput2* was developed by Makoto Ishisone, SRA. For details, contact SRA, Japan.

6. Printer Filters

The primary localization function of filtering of print files is to modify the character set encoding in the original file to match (or better match) the character sets available on a given printing device. Filtering becomes necessary when the printing device does not have the character set encoding that is necessary for the file being printed. Filtering is not the appropriate approach for matching file type to print device if the print device allows downloading of fonts and the appropriate font exists. In that case, the matching font should be downloaded to the print device.

The System V print service allows for the definition of filters that can be used to transform one type of input to another. The job submission command (lp) allows the user to specify the character sets of the submitted print job. This specification will be passed along to filters and the printer interface. You should provide at least one filter that is prepared to intercept and use this character set information for inclusion into any filter chains.

6.1 Using STREAMS kbd Maps

A particularly useful filter implementation is one which uses the kbd map tables for character translation information. Such a filter can be based on the *kbdpipe* command or directly implemented in an application filter. Alternatively, the printer interface can push the kbd streams module and directly use the kbd map tables upon output to the printer.

See Chapter 4, 7.

6.2 Other Methods and Considerations

The following considerations should be applied in determining how the localization package will provide character set translation on output.

- Where the printer in use allows downloading of fonts or modification of character set coding on the printer, that method is almost always superior to filtering. It will usually call for a printer interface specific to the printer, that understands how to perform the appropriate operations to obtain the character set desired.
- STREAMS module based translation will generally be faster than a filter.
- The standard System V printer interfaces do not provide for pushing streams modules and loading kbd maps. Your package can provide modified interface scripts to push STREAMS modules and initialize kbd maps. Alternatively, you can provide these two functions in a print service filter.
- Site specific, locally customized interface scripts probably do not provide the required STREAMS support. In that case filters are preferred to STREAMS based kbd translation tables.
- Generally, format conversion filters should be provided as print service filters. Character set translation filters are a special case of format conversion and are most appropriately provided as filters unless direct access to the printer device is necessary (e.g., pushing streams modules.)

- The ordering of filtering and translation steps may be important. For example, a simple text file, printed on a PostScript printer, should have character set filtering applied before being converted to PostScript format.

Refer to

- Chapter 9, titled "Print Service" in the *SVR4 System Administrator's Guide*
- *terminfo*(4) in the *SVR4 System Administrator's Reference Manual*

for more information.

7. System Administration

This chapter outlines the steps that the System Administrator must take to activate a localized system. Installation of localized packages and startup procedures are explained in this chapter.

7.1 Installation

System V provides tools to help automate software packaging, installation, and de-installation procedures. It is strongly recommended to use packaging tools to manage localization files or packages.

The following commands are the most frequently used packaging commands.

— *pkgmk* command produces an installable package to be used as input to the *pkgadd* command.

— *pkgadd* command transfers the contents of a software package from the distribution medium or directory to install it onto the system.

— *pkginfo* command displays information about software packages which are installed on the system or which reside on a particular device or directory.

— *pkgrm* command will remove a previously installed or partially installed package from the system.

Refer to

- Software Management of System Administrator's Guide
- *pkgadd*(1M), *pkgask*(1M), *pkgchk*(1M), *pkgrm*(1M), and *pkgtrans*(1)

for more information.

7.2 Startup Procedures

7.2.1 rc-files

The directory /sbin/rc{n}.d contains files executed by /usr/sbin/rc{n} for transitions to system states {n}. Files in this directory are linked from the /sbin/init.d directory, and begin with either **K** or **S**. **K** shows processes that are stopped, and **S** shows processes that are started when entering system states {n}.

To stop or start any locale specific processes at system start up time, the appropriate files in rc{n}.d directory should be modified.

Refer to

— *rc0*(1M) (for stopping the operating system)

— *rc2*(1M) (for multi-user environment startup)

— *rc6*(1M) (for stopping and rebooting the operating system)

in the *SVR4 System Administrator's Reference Manual* for more information.

7.3 Time Zone setting

The time zone setting for a user is generally set on a system-wide basis. The environment variable "**TZ**" designates the offset from the machine UTC time zone, and establishes the time zone that a user expects. Several System V utilities use the **TZ** variable in addition to the date and time formats that are described by the data file designated by the **"LC_TIME"** variable.

In System V, there are two time zone value formats:

- POSIX time zone standard format
- System V time zone extension format

Refer to

- Section 8, titled "Language-Specific Services for the C Programming Language" in the *POSIX 1003.1*
- Section 8, titled "Language-Specific Services for the C Programming Language" in the *SVR4 Programmer's Guide: POSIX Conformance*
- *timezone*(4), *environ*(5) in the *SVR4 System Administrator's Reference Manual*

for more information.

7.4 System and User Profile setting

The system performs several routines between the time a user logs in and the time a shell prompt appears. These routines are listed as two files that are executed during the login sequence: the system profile (/etc/profile) and the user profile ($HOME/.profile).

These two files need to be changed to serve the needs of a localized system.

— Define and export locale specific environment variables.

— Execute locale specific programs.

— Display any locale specific messages.

Refer to

- *profile*(4) and *environ*(5) in the *SVR4 System Administrator's Reference Manual*

for more information.

7.5 STREAMS configuration

7.5.1 Putting the kbd module into current tty stream

Before doing any operations with the *kbd* module, it must be attached to the current tty stream. strconf(1) is used to do this. If there is no *kbd* module in the tty stream, strchg(1) is used to push the *kbd* module into the tty stream. The *kbd* module must be pushed below the *ldterm* module, because the *ldterm* assumes that its input and output characters are in EUC, and canonical mode processing of the ldterm depends upon that fact. Since there are some modules above the *ldterm*, every module up to and including the *ldterm*, must be popped, before the *kbd* module is pushed into the stream. The remaining modules may then be pushed back into the stream. The stty settings of the tty stream must be saved since current settings are discarded when the ldterm is popped. After pushing all the STREAMS modules, saved stty settings may be restored using the stty(1) command. eucset(1) must be called to set up the byte and column width of each codeset.

Refer to

- *strconf*(1) and *strchg*(1) in the *SVR4 User's Reference Manual*,
- *eucset*(1) in the *MNLS 4.0 Product Overview*

for more detailed information.

7.5.2 Load the tables into kernel

For the *kbd* module to access conversion tables, they must be loaded into kernel memory space. kbdload(1M) is used to load conversion tables into kernel memory space. Conversion tables are loaded into kernel memory space in two states, *private* or *public*. To debug conversion tables, users may load them privately and check if the conversion is done correctly.

Refer to *kbdload*(1M) in the *MNLS 4.0 Product Overview* for more details.

7.6 Select table used for input and output conversion

To use conversion tables, the table to be used for input conversion of the tty stream and the table to be used for output are selected by using kbdset(1).

For example, to select the *stou* module for input conversion, type:

    kbdset stou

and to select the *utos* module for output conversion, type:

    kbdset -o utos

Refer to

- *autopush*(1M),*strch*(1),*strchg*(1) in the *SVR4 Programmer's Guide : STREAMS*

for more information.

Appendix A: Locale Names Used in ELS and Other Products

The following are tables of locale names in use by USL's System V Release 4.0 European Language Supplement (ELS) and other products. These locales are not provided as part of SVR4.1 ES.

Locale	Language	Territory	Code-set
da_DK	Danish	Denmark	ISO8859-1
da_DK.850	Danish	Denmark	PC850
da_DK865	Danish	Denmark	PC865
nl_NL	Dutch	Netherlands	ISO8859-1
nl_NL.437	Dutch	Netherlands	PC437
nl_NL.850	Dutch	Netherlands	PC850
nl_BE	Dutch	Belgium	ISO8859-1
nl_BE.437	Dutch	Belgium	PC437
nl_BE.850	Dutch	Belgium	PC850
en_GB	English	United Kingdom	ISO8859-1
en_GB.646	English	United Kingdom	ISO646
en_GB.437	English	United Kingdom	PC437
en_GB.850	English	United Kingdom	PC850
en_AU	English	Australia	ISO8859-1
en_AU.646	English	Australia	ISO646
en_AU.437	English	Australia	PC437
en_AU.850	English	Australia	PC850
en_CA	English	Canada	ISO8859-1
en_CA.646	English	Canada	ISO646
en_CA.437	English	Canada	PC437
en_CA.850	English	Canada	PC850
en_US	English	United States	ISO8859-1
en_US.646	English	United States	ISO646
en_US.437	English	United States	PC437
en_US.850	English	United States	PC850
fi_FI	Finnish	Finland	ISO8859-1
fi_FI.437	Finnish	Finland	PC437
fi_FI.850	Finnish	Finland	PC850
fr_FR	French	France	ISO8859-1
fr_FR.437	French	France	PC437
fr_FR.850	French	France	PC850
fr_BE	French	Belgium	ISO8859-1
fr_BE.437	French	Belgium	PC437
fr_BE.850	French	Belgium	PC850
fr_CA	French	Canada	ISO8859-1

Locale	Language	Territory	Code-set
fr_CA.850	French	Canada	PC850
fr_CA.863	French	Canada	PC863
fr_CH	French	Switzerland	ISO8859-1
fr_CH.437	French	Switzerland	PC437
fr_CH.850	French	Switzerland	PC850
de_DE	German	Germany	ISO8859-1
de_DE.437	German	Germany	PC437
de_DE.850	German	Germany	PC850
de_AT	German	Austria	ISO8859-1
de_AT.437	German	Austria	PC437
de_AT.850	German	Austria	PC850
de_CH	German	Switzerland	ISO8859-1
de_CH.437	German	Switzerland	PC437
de_CH.850	German	Switzerland	PC850
el_GR	Greek	Greece	ISO8859-7
is_IS	Icelandic	Iceland	ISO8859-1
is_IS.850	Icelandic	Iceland	PC850
it_IT	Italian	Italy	ISO8859-1
it_IT.437	Italian	Italy	PC437
it_IT.850	Italian	Italy	PC850
it_CH	Italian	Switzerland	ISO8859-1
it_CH.437	Italian	Switzerland	PC437
it_CH.850	Italian	Switzerland	PC850
no_NO	Norwegian	Norway	ISO8859-1
no_NO.850	Norwegian	Norway	PC850
no_NO.865	Norwegian	Norway	PC865
pl_PL	Polish	Poland	ISO8859-2
pt_PT	Portuguese	Portugal	ISO8859-1
pt_PT.850	Portuguese	Portugal	PC850
pt_PT.860	Portuguese	Portugal	PC860
ru_SU	Russian	USSR	ISO8859-5
sh_YU	SerboCroation	Yugoslavia	ISO8859-2
sk_CS	Slovak	Czechoslovakia	ISO8859-2
es_ES	Spanish	Spain	ISO8859-1
es_ES.437	Spanish	Spain	PC437
es_ES.850	Spanish	Spain	PC850
sv_SE	Swedish	Sweden	ISO8859-1
sv_SE.437	Swedish	Sweden	PC437
sv_SE.850	Swedish	Sweden	PC850
tr_TR	Turkish	Turkey	ISO8859-3

Locale	Language	Territory	Code-set	Encoding Scheme
ar	Arabic	Arab countries	ISO8859-6	
eu	Basque	W Pyrenees	ISO8859-1	
bg	Bulgarian	Bulgaria	ISO8859-5	
ca	Catalan	Catalonia	ISO8859-3	
china	Chinese	MainlandChina	GB2312-80	EUC
chinese	Chinese	Taiwan,R.O.C,	ASCII,CNS 11643	EUC
chinese	Chinese	Mainland China	ASCII,GB-2312	EUC
tchinese	Chinese	Taiwan,R.O.C.	ASCII,CNS 11643	EUC
zh_TW.eucTW	Chinese	Taiwan,R.O.C.	ASCII,CNS 11643	EUC
zh_TW.TCA	Chinese	Taiwan,R.O.C.	ASCII,CNS 11643	TCA
zh_TW	Chinese	Taiwan,R.O.C.	ASCII,CNS 11643	EUC
zh	Chinese	Mainland China	ASCII,GB 2312	EUC
co	Corsican	Corsica	ISO8859-1	
hr	Croatian	Croatia	ISO8859-2	
cs	Czech	Czechoslovakia	ISO8859-2	
da	Danish	Denmark	ISO8859-1	
da_DK.88591	Danish	Denmark	ISO8859-1	
nl_BE.88591	Dutch	Belgium	ISO8859-1	
nl	Dutch	Netherlands	ISO8859-1	
nl_NL.88591	Dutch	Netherlands	ISO8859-1	
En_US	English	U.S.A	ASCII	
En_GB	English	United Kingdom	ASCII	
en_GB.88591	English	United Kingdom	ISO8859-1	
en_UK	English	United Kingdom	ISO8859-1	
en_US.88591	English	United States	ISO8859-1	
en_US	English	United States	ASCII	
C	English	UNIX	ASCII	
en	English	varies	ASCII	
eo	Esperanto	worldwide	ISO8859-3	
et	Estonian	Estonia	ISO8859-4	
fi	Finnish	Finland	ISO8859-4	
fi_FI.88591	Finnish	Finland	ISO8859-1	
fr_BE.88591	French	Belgium	ISO8859-1	
fr_BE	French	Belgium	ISO8859-1	
fr_CA	French	Canada	ISO8859-1	
fr_CA.88591	French	Cananda	ISO8859-1	
fr	French	France	ISO8859-1	
fr_FR.88591	French	France	ISO8859-1	
fr_CH.88591	French	Switzerland	ISO8859-1	
fr_CH	French	Switzland	ISO8859-1	

Locale	Language	Territory	Code-set	Encoding Scheme
fy	Frisian	Frisia	ISO8859-1	
ga	Irish	Ireland	ISO8859-1	
gl	Galician	Galicia	ISO8859-3	
de_BE.88591	German	Belgium	ISO8859-1	
de	German	Germany	ISO8859-1	
de_DE.88591	German	Germany	ISO8859-1	
de_CH.88591	German	Switzerland	ISO8859-1	
de_CH	German	Switzerland	ISO8859-1	
el	Greek	Greece	ISO8859-7	
hu	Hungarian	Hungary	ISO8859-2	
indian_hin	Hindi	India	ISCII	EUC
is	Icelandic	Iceland	ISO8859-1	
iw	Hebrew	Israel	ISO8859-8	
it	Italian	Italy	ISO8859-1	
it_IT.88591	Italian	Italy	ISO8859-1	
ja	Japanese	Japan	JIS X0201,X0208	EUC
ja_JP.EUC	Japanese	Japan	JIS X0201,X0208, X0212	EUC
ja_JP	Japanese	Japan	JIS X0201,X0208	
japanese	Japanese	Japan	JIS X0201,X0208	EUC
ja_JP.SJIS	Japanese	Japan	JIS X0201,X0208	MS-Kanji
japan	Japanese	Japan	JIS X0201,X0208	EUC
kl	Greenlandic	Greenland	ISO8859-1	
ko_KR.5601	Korean	Korea	ASCII,KSC5601	EUC
ko_KR.EUC	Korean	Korea	ASCII,KSC5601	EUC
korea	Korean	Korea	KS C 5601	EUC
korean	Korean	Korea	ASCII,KSC5601	EUC
ko	Korean	Korea	ASCII,KSC5601	EUC
lt	Lithuanian	Lithuania	ISO8859-4	
lv	Latvian	Latvia	ISO8859-4	
mt	Maltese	Malta	ISO8859-3	
no	Norwegian	Norway	ISO8859-1	
no_NO.88591	Norwegian	Norway	ISO8859-1	
fa	Persian	Iran	ISO8859-6	
pl	Polish	Poland	ISO8859-2	
pt_BR	Portuguese	Brazil	ISO8859-1	
pt	Portuguese	Portugal	ISO8859-1	
pt_PT.88591	Portuguese	Portugal	ISO8859-1	
ro	Romanian	Romania	ISO8859-2	
ru	Russian	Soviet Union	ISO8859-5	

Locale	Language	Territory	Code-set	Encoding Scheme
gd	Scots Gaelic	Scotland	ISO8859-1	
sr	Serbian	Serbia	ISO8859-5	
sh	Serbo-Croatian	Yugoslavia	ISO8859-2 or 8859-5	
sk	Slovak	Czechoslovakia	ISO8859-2	
sl	Slovene	Slovenia	ISO8859-2	
es_ES.88591	Spanish	Spain	ISO8859-1	
es	Spanish	Spain, etc.	ISO8859-1	
sv	Swedish	Sweden	ISO8859-1	
sv_SE.88591	Swedish	Sweden	ISO8859-1	
tr	Turkish	Turkey	ISO8859-5	
uk	Ukranian	Ukrania	ISO8859-5	
cy	Welsh	Wales	ISO8859-1	
ji	Yiddish	diaspora	ISO8859-1	

Appendix B: ISO 639:1988 Two-letter language symbols

These come from *ISO 639:1988 Code for the representation of names of languages* **Alphabetical list of two-letter language symbols**. For more detailed information about the original names of the two-letter lanugage symbols, please refer to ISO 639:1988.

Language symbol	Language name
aa	Afar
ab	Abkhazian
af	Afrikaans
am	Amharic
ar	Arabic
as	Assamese
ay	Aymara
az	Azerbaijani
ba	Bashkir
be	Byelorussian
bg	Bulgarian
bh	Bihari
bi	Bislama
bn	Bengali; Bangla
bo	Tibetan [Bodskad]
br	Breton
ca	Catalan
co	Corsican
cs	Czech
cy	Welsh [Cymraeg]
da	Danish
de	German
dz	Bhutani [Dzongkha]
el	Greek [Ellinika]
en	English
eo	Esperanto
es	Spanish [Espan ol]
et	Estonian
eu	Basque [Euskera]
fa	Persian [Farsi]
fi	Finnish
fj	Fiji
fo	Faroese
fr	French
fy	Frisian

Appendix B: ISO 639:1988 Two-letter language symbols

Language symbol	Language name
ga	Irish [Gaeilge]
gd	Scots Gaelic [Gaidhilge]
gl	Galician
gn	Guarani
gu	Gujarati
ha	Hausa
hi	Hindi
hr	Croatian [Hrvatski]
hu	Hungarian
hy	Armenian [Hayeren]
ia	Interlingua
ie	Interlingue
ik	Inupiak
in	Indonesian
is	Icelandic
it	Italian
iw	Hebrew [Iwrith]
ja	Japanese
ji	Yiddish
ji	Yiddish [Jiddisch]
jw	Javanese [Bahasa Jawa]
ka	Georgian [Kartuli]
kk	Kazakh
kl	Greenlandic [Kalaallisut]
km	Cambodian [Khmer]
kn	Kannada
ko	Korean
ks	Kashmiri
ku	Kurdish
ky	Kirghiz [Kyrgyz]
la	Latin
ln	Lingala
lo	Laothian
lt	Lithuanian
lv	Latvian, Lettish
mg	Malagasy
mi	Maori
mk	Macedonian
ml	Malayalam
mn	Mongolian
mo	Moldavian

Language symbol	Language name
mr	Marathi
ms	Malay
mt	Maltese
my	Burmese [Myanmasa]
na	Nauru
ne	Nepali
nl	Dutch [Nederlands]
no	Norwegian
oc	Occitan
om	(Afan) Oromo
or	Oriya
pa	Punjabi [Panjabi]
pl	Polish
ps	Pashto, Pushto
pt	Portuguese
qu	Quechua
rm	Rhaeto-Romance
rn	Kirundi
ro	Romanian
ru	Russian
rw	Kinyarywanda
sa	Sanskrit
sd	Sindhi
sg	Sangho
sh	Serbo-Croatian
si	Singhalese
sk	Slovak
sl	Slovenian
sm	Samoan
sn	Shona
so	Somali
sq	Albanian [Shqip]
sr	Serbian
ss	Siswati
st	Sesotho
su	Sundanese
sv	Swedish [Svenska]
sw	Swahili
ta	Tamil
te	Telugu
tg	Tajik

Language symbol	Language name
th	Thai
ti	Tigrinya
tk	Turkmen
tl	Tagalog
tn	Setswana
to	Tonga
tr	Turkish
ts	Tsonga
tt	Tatar
tw	Twi
uk	Ukrainian
ur	Urdu
uz	Uzbek
vi	Vietnamese
vo	Volapuk
wo	Wolof
xh	Xhosa
yo	Yoruba
zh	Chinese [Zhongwen]
zu	Zulu

Appendix C: ISO 3166:1988 Two-letter country code

These come from *ISO 3166:1988 Code for the representation of name of countries*

ENTITY (short name in English)	Alpha-2 code *
AFGHANISTAN	AF
ALBANIA	AL
ALGERIA	DZ
AMERICAN SAMOA	AS
ANDORRA	AD
ANGOLA	AO
ANGUILLA	AI
ANTARCTICA	AQ
ANTIGUA AND BARBUDA	AG
ARGENTINA	AR
ARUBA	AW
AUSTRALIA	AU
AUSTRIA	AT
BAHAMAS	BS
BAHRAIN	BH
BANGLADESH	BD
BARBADOS	BB
BELGIUM	BE
BELIZE	BZ
BENIN	BJ
BERMUDA	BM
BHUTAN	BT
BOLIVIA	BO
BOTSWANA	BW
BOUVET ISLAND	BV
BRAZIL	BR
BRITISH INDIAN OCEAN TERRITORY	IO
BRUNEI DARUSSALAM	BN
BULGARIA	BG
BURKINA FASO	BF
BURMA	BU
BURUNDI	BI
BYELORUSSIAN SSR	BY
CAMEROON	CM
CANADA	CA
CAPE VERDE	CV
CAYMAN ISLANDS	KY

ENTITY (short name in English)	Alpha-2 code *
CENTRAL AFRICAN REPUBLIC	CF
CHAD	TD
CHILE	CL
CHINA	CN
CHRISTMAS ISLAND	CX
COCOS (KEELING) ISLAND	CC
COLOMBIA	CO
COMOROS	KM
CONGO	CG
COOK ISLANDS	CK
COSTA RICA	CR
COTE D’IVOIRE	CI
CUBA	CU
CYPRUS	CY
CZECHOSLOVAKIA	CS
DENMARK	DK
DJIBOUTI	DJ
DOMINICA	DM
DOMINICAN REPUBLIC	DO
EAST TIMOR	TP
ECUADOR	EC
EGYPT	EG
EL SALVADOR	SV
EQUATORIAL GUINEA	GQ
ETHIOPIA	ET
FALKLAND ISLANDS (MALVINAS)	FK
FAROE ISLANDS	FO
FIJI	FJ
FINLAND	FI
FRANCE	FR
FRENCH GUIANA	GF
FRENCH POLYNESIA	PF
FRENCH SOUTHERN TERRITORIES	TF
GABON	GA
GAMBIA	GM
GERMAN DEMOCRATIC REPUBLIC	DD
GERMANY, FEDERAL REPUBLIC OF	DE
GHANA	GH
GIBRALTAR	GI
GREECE	GR
GREENLAND	GL

ENTITY (short name in English)	Alpha-2 code *
GRENADA	GD
GUADELOUPE	GP
GUAM	GU
GUATEMALA	GT
GUINEA	GN
GUINEA-BISSAU	GW
GUYANA	GY
HAITI	HT
HEARD AND MC DONALD ISLANDS	HM
HONDURAS	HN
HONG KONG	HK
HUNGARY	HU
ICELAND	IS
INDIA	IN
INDONESIA	ID
IRAN (ISLAMIC REPUBLIC OF)	IR
IRAQ	IQ
IRELAND	IE
ISRAEL	IL
ITALY	IT
JAMAICA	JM
JAPAN	JP
JORDAN	JO
KAMPUCHEA, DEMOCRATIC	KH
KENYA	KE
KIRIBATI	KI
KOREA, DEMOCRATIC PEOPLE’S REPUBLIC OF	KP
KOREA, REPUBLIC OF	KR
KUWAIT	KW
LAO PEOPLE’S DEMOCRATIC	LA
LEBANON	LB
LESOTHO	LS
LIBERIA	LR
LIBYAN ARAB JAMAHIRIYA	LY
LIECHTENSTEIN	LI
LUXEMBOURG	LU
MACAU	MO
MADAGASCAR	MG
MALAWI	MW
MALAYSIA	MY
MALDIVES	MV

Appendix C: ISO 3166:1988 Two-letter country code

ENTITY (short name in English)	Alpha-2 code *
MALI	ML
MALTA	MT
MARSHALL ISLANDS	MH
MARTINIQUE	MQ
MAURITANIA	MR
MAURITIUS	MU
MEXICO	MX
MICRONESIA	FM
MONACO	MC
MONGOLIA	MN
MONTSERRAT	MS
MOROCCO	MA
MOZAMBIQUE	MZ
NAMIBIA	NA
NAURU	NR
NEPAL	NP
NETHERLANDS	NL
NETHERLANDS ANTILLES	AN
NEUTRAL ZONE	NT
NEW CALEDONIA	NC
NEW ZEALAND	NZ
NICARAGUA	NI
NIGER	NE
NIGERIA	NG
NIUE	NU
NORFOLK ISLAND	NF
NORTHERN MARIANA ISLAND	MP
NORWAY	NO
OMAN	OM
PAKISTAN	PK
PALAU	PW
PANAMA	PA
PAPUA NEW GUINEA	PG
PARAGUAY	PY
PERU	PE
PHILIPPINES	PH
PITCAIRN	PN
POLAND	PL
PORTUGAL	PT
PUERTO RICO	PR
QATAR	QA

ENTITY (short name in English)	Alpha-2 code *
REUNION	RE
ROMANIA	RO
RWANDA	RW
ST. HELENA	SH
SAINT KITTS AND NEVIS	KN
SAINT LUCIA	LC
ST. PIERRE AND MIQUELON	PM
SAINT VINCENT AND THE GRENADINES	VC
SAMOA	WS
SAN MARINO	SM
SAO TOME AND PRINCIPE	ST
SAUDI ARABIA	SA
SENEGAL	SN
SEYCHELLES	SC
SIERRA LEONE	SL
SINGAPORE	SG
SOLOMON ISLANDS	SB
SOMALIA	SO
SOUTH AFRICA	ZA
SPAIN	ES
SRI LANKA	LK
SUDAN	SD
SURINAME	SR
SVALBARD AND JAN MAYEN ISLANDS	SJ
SWAZILAND	SZ
SWEDEN	SE
SWITZERLAND	CH
SYRIAN ARAB REPUBLIC	SY
TAIWAN, PROVINCE OF CHINA	TW
TANZANIA, UNITED REPUBLIC OF	TZ
THAILAND	TH
TOGO	TG
TOKELAU	TK
TONGA	TO
TRINIDAD AND TOBAGO	TT
TUNISIA	TN
TURKEY	TR
TURKS AND CAICOS ISLANDS	TC
TUVALU	TV
UGANDA	UG
UKRAINIAN SSR	UA

ENTITY (short name in English)	Alpha-2 code *
UNITED ARAB EMIRATES	AE
UNITED KINGDOM	GB
UNITED STATES	US
UNITED STATES MINOR OUTLYING ISLANDS	UM
URUGUAY	UY
USSR	SU
VANUATU	VU
VATICAN CITY STATE (HOLY SEE)	VA
VENEZUELA	VE
VIET NAM	VN
VIRGIN ISLANDS (BRITISH)	VG
VIRGIN ISLANDS (U.S.)	VI
WALLIS AND FUTUNA ISLANDS	WF
WESTERN SAHARA	EH
YEMEN	YE
YEMEN, DEMOCRATIC	YD
YUGOSLAVIA	YU
ZAIRE	ZR
ZAMBIA	ZM
ZIMBABWE	ZW

Appendix D: LC_CTYPE and LC_NUMERIC Example

The following is an example of an input file used to create the Japanese character classification and conversion definitions tables and store them in files designated by the LC_CTYPE and LC_NUMERIC, respectively.

```
#
# locale JAPAN
#
LC_CTYPE    LC_CTYPE
#
# specification for single byte characters
#
isupper       0x41 - 0x5a
islower       0x61 - 0x7a
isdigit       0x30 - 0x39
isspace       0x20   0x9 - 0xd
ispunct       0x21 - 0x2f     0x3a - 0x40  \
              0x5b - 0x60     0x7b - 0x7e
iscntrl       0x0 - 0x1f      0x7f - 0x9f
isblank       0x20
isxdigit  0x30 - 0x39   0x61 - 0x66 0x41 - 0x46
ul          <0x41 0x61> <0x42 0x62> <0x43 0x63> \
            <0x44 0x64> <0x45 0x65> <0x46 0x66> \
            <0x47 0x67> <0x48 0x68> <0x49 0x69> \
            <0x4a 0x6a> <0x4b 0x6b> <0x4c 0x6c> \
            <0x4d 0x6d> <0x4e 0x6e> <0x4f 0x6f> \
            <0x50 0x70> <0x51 0x71> <0x52 0x72> \
            <0x53 0x73> <0x54 0x74> <0x55 0x75> \
            <0x56 0x76> <0x57 0x77> <0x58 0x78> \
            <0x59 0x79> <0x5a 0x7a>
cswidth           2:2,1:1,2:2
LC_NUMERIC  LC_NUMERIC
decimal_point     .
thousands_sep     ,
#
# specification for supplementary code-set 1
#
LC_CTYPE1
isupper           0xa3c1 - 0xa3da
islower           0xa3e1 - 0xa3fa
isdigit           0xa3b0 - 0xa3b9
isspace           0xa1a1
isphonogram 0xa4a1 - 0xa4f3 0xa5a1 - 0xa5f6
isideogram  0xb0a1 - 0xb0fe 0xb1a1 - 0xb1fe 0xb2a1 - 0xb2fe \
            0xb3a1 - 0xb3fe 0xb4a1 - 0xb4fe 0xb5a1 - 0xb5fe \
            0xb6a1 - 0xb6fe 0xb7a1 - 0xb7fe 0xb8a1 - 0xb8fe \
            0xb9a1 - 0xb9fe 0xbaa1 - 0xbafe 0xbba1 - 0xbbfe \
            0xbca1 - 0xbcfe 0xbda1 - 0xbdfe 0xbea1 - 0xbefe \
            0xbfa1 - 0xbffe 0xc0a1 - 0xc0fe 0xc1a1 - 0xc1fe \
            0xc2a1 - 0xc2fe 0xc3a1 - 0xc3fe 0xc4a1 - 0xc4fe \
            0xc5a1 - 0xc5fe 0xc6a1 - 0xc6fe 0xc7a1 - 0xc7fe \
            0xcca1 - 0xccfe 0xcda1 - 0xcdfe 0xcea1 - 0xcefe \
            0xcfa1 - 0xcffe 0xd0a1 - 0xd0fe 0xd1a1 - 0xd1fe \
```

```
             0xd2a1 - 0xd2fe 0xd3a1 - 0xd3fe 0xd4a1 - 0xd4fe \
             0xd5a1 - 0xd5fe 0xd6a1 - 0xd6fe 0xd7a1 - 0xd7fe \
             0xd8a1 - 0xd8fe 0xd9a1 - 0xd9fe 0xdaa1 - 0xdafe \
             0xdba1 - 0xdbfe 0xdca1 - 0xdcfe 0xdda1 - 0xddfe \
             0xdea1 - 0xdefe 0xdfa1 - 0xdffe 0xe0a1 - 0xe0fe \
             0xe1a1 - 0xe1fe 0xe2a1 - 0xe2fe 0xe3a1 - 0xe3fe \
             0xe4a1 - 0xe4fe 0xe5a1 - 0xe5fe 0xe6a1 - 0xe6fe \
             0xe7a1 - 0xe7fe 0xe8a1 - 0xe8fe 0xe9a1 - 0xe9fe \
             0xeaa1 - 0xeafe 0xeba1 - 0xebfe 0xeca1 - 0xecfe \
             0xeda1 - 0xedfe 0xeea1 - 0xeefe 0xefa1 - 0xeffe \
             0xf0a1 - 0xf0fe 0xf1a1 - 0xf1fe 0xf2a1 - 0xf2fe \
             0xf3a1 - 0xf3fe 0xf4a1 - 0xf4fe 0xf5a1 - 0xf5fe \
             0xf6a1 - 0xf6fe 0xf7a1 - 0xf7fe 0xf8a1 - 0xf8fe \
             0xf9a1 - 0xf9fe 0xfaa1 - 0xfafe 0xfba1 - 0xfbfe \
             0xfca1 - 0xfcfe 0xfda1 - 0xfdfe 0xfea1 - 0xfefe \
isenglish    0xa3c1 - 0xa3da 0xa3e1 - 0xa3fa
isnumber     0xa3b0 - 0xa3b9
isspecial    0xa1a2 - 0xa1fe 0xa2a1 - 0xa2ae 0xa2ba - 0xa2c1 \
             0xa2ca - 0xa2d0 0xa2dc - 0xa2ea 0xa2f2 - 0xa2f9 \
             0xa2fe
iswchar6     0xa6a1 - 0xa6b8 0xa6c1 - 0xa6d8 0xa7a1 - 0xa7c1 \
             0xa7d1 - 0xa7f1
#
#            JIS X0208 whole code-set
#
iswchar9     0xa1a1 - 0xa1fe 0xa2a1 - 0xa2fe 0xa3a1 - 0xa3fe \
             0xa4a1 - 0xa4fe 0xa5a1 - 0xa5fe 0xa6a1 - 0xa6fe \
             0xa7a1 - 0xa7fe 0xa8a1 - 0xa8fe 0xa9a1 - 0xa9fe \
             0xaaa1 - 0xaafe 0xaba1 - 0xabfe 0xaca1 - 0xacfe \
             0xada1 - 0xadfe 0xaea1 - 0xaefe 0xafa1 - 0xaffe \
             0xb0a1 - 0xb0fe 0xb1a1 - 0xb1fe 0xb2a1 - 0xb2fe \
             0xb3a1 - 0xb3fe 0xb4a1 - 0xb4fe 0xb5a1 - 0xb5fe \
             0xb6a1 - 0xb6fe 0xb7a1 - 0xb7fe 0xb8a1 - 0xb8fe \
             0xb9a1 - 0xb9fe 0xbaa1 - 0xbafe 0xbba1 - 0xbbfe \
             0xbca1 - 0xbcfe 0xbda1 - 0xbdfe 0xbea1 - 0xbefe \
             0xbfa1 - 0xbffe 0xc0a1 - 0xc0fe 0xc1a1 - 0xc1fe \
             0xc2a1 - 0xc2fe 0xc3a1 - 0xc3fe 0xc4a1 - 0xc4fe \
             0xc5a1 - 0xc5fe 0xc6a1 - 0xc6fe 0xc7a1 - 0xc7fe \
             0xc8a1 - 0xc8fe 0xc9a1 - 0xc9fe 0xcaa1 - 0xcafe \
             0xcba1 - 0xcbfe 0xcca1 - 0xccfe 0xcda1 - 0xcdfe \
             0xcea1 - 0xcefe 0xcfa1 - 0xcffe 0xd0a1 - 0xd0fe \
             0xd1a1 - 0xd1fe 0xd2a1 - 0xd2fe 0xd3a1 - 0xd3fe \
             0xd4a1 - 0xd4fe 0xd5a1 - 0xd5fe 0xd6a1 - 0xd6fe \
             0xd7a1 - 0xd7fe 0xd8a1 - 0xd8fe 0xd9a1 - 0xd9fe \
             0xdaa1 - 0xdafe 0xdba1 - 0xdbfe 0xdca1 - 0xdcfe \
             0xdda1 - 0xddfe 0xdea1 - 0xdefe 0xdfa1 - 0xdffe \
             0xe0a1 - 0xe0fe 0xe1a1 - 0xe1fe 0xe2a1 - 0xe2fe \
             0xe3a1 - 0xe3fe 0xe4a1 - 0xe4fe 0xe5a1 - 0xe5fe \
             0xe6a1 - 0xe6fe 0xe7a1 - 0xe7fe 0xe8a1 - 0xe8fe \
             0xe9a1 - 0xe9fe 0xeaa1 - 0xeafe 0xeba1 - 0xebfe \
             0xeca1 - 0xecfe 0xeda1 - 0xedfe 0xeea1 - 0xeefe \
             0xefa1 - 0xeffe 0xf0a1 - 0xf0fe 0xf1a1 - 0xf1fe \
             0xf2a1 - 0xf2fe 0xf3a1 - 0xf3fe 0xf4a1 - 0xf4fe \
             0xf5a1 - 0xf5fe 0xf6a1 - 0xf6fe 0xf7a1 - 0xf7fe \
             0xf8a1 - 0xf8fe 0xf9a1 - 0xf9fe 0xfaa1 - 0xfafe \
```

```
            0xfba1 - 0xfbfe 0xfca1 - 0xfcfe 0xfda1 - 0xfdfe \
            0xfea1 - 0xfefe
#
#           JIS X0208 parentheses
#
iswchar10   0xa1c6 - 0xa1db
#
#           JIS X0208 hiragana
#
iswchar11   0xa4a1 - 0xa4f3
#
#           JIS X0208 katakana
#
iswchar12   0xa5a1 - 0xa5f6
#
#           JIS X0208 other characters
#
iswchar13   0xa6a1 - 0xa6b8 0xa6c1 - 0xa6d8 0xa7a1 - 0xa7c1 \
            0xa7d1 - 0xa7f1 0xa8a1 - 0xa8bf
#
#           English letter translation table
#
ul          <0xa3c1 0xa3e1> <0xa3c2 0xa3e2> <0xa3c3 0xa3e3> \
            <0xa3c4 0xa3e4> <0xa3c5 0xa3e5> <0xa3c6 0xa3e6> \
            <0xa3c7 0xa3e7> <0xa3c8 0xa3e8> <0xa3c9 0xa3e9> \
            <0xa3ca 0xa3ea> <0xa3cb 0xa3eb> <0xa3cc 0xa3ec> \
            <0xa3cd 0xa3ed> <0xa3ce 0xa3ee> <0xa3cf 0xa3ef> \
            <0xa3d0 0xa3f0> <0xa3d1 0xa3f1> <0xa3d2 0xa3f2> \
            <0xa3d3 0xa3f3> <0xa3d4 0xa3f4> <0xa3d5 0xa3f5> \
            <0xa3d6 0xa3f6> <0xa3d7 0xa3f7> <0xa3d8 0xa3f8> \
            <0xa3d9 0xa3f9> <0xa3da 0xa3fa> \
#
#           kana translation table
#
            <0xa4a1 0xa5a1> <0xa4a2 0xa5a2> <0xa4a3 0xa5a3> \
            <0xa4a4 0xa5a4> <0xa4a5 0xa5a5> <0xa4a6 0xa5a6> \
            <0xa4a7 0xa5a7> <0xa4a8 0xa5a8> <0xa4a9 0xa5a9> \
            <0xa4aa 0xa5aa> <0xa4ab 0xa5ab> <0xa4ac 0xa5ac> \
            <0xa4ad 0xa5ad> <0xa4ae 0xa5ae> <0xa4af 0xa5af> \
            <0xa4b0 0xa5b0> <0xa4b1 0xa5b1> <0xa4b2 0xa5b2> \
            <0xa4b3 0xa5b3> <0xa4b4 0xa5b4> <0xa4b5 0xa5b5> \
            <0xa4b6 0xa5b6> <0xa4b7 0xa5b7> <0xa4b8 0xa5b8> \
            <0xa4b9 0xa5b9> <0xa4ba 0xa5ba> <0xa4bb 0xa5bb> \
            <0xa4bc 0xa5bc> <0xa4bd 0xa5bd> <0xa4be 0xa5be> \
            <0xa4bf 0xa5bf> <0xa4c0 0xa5c0> <0xa4c1 0xa5c1> \
            <0xa4c2 0xa5c2> <0xa4c3 0xa5c3> <0xa4c4 0xa5c4> \
            <0xa4c5 0xa5c5> <0xa4c6 0xa5c6> <0xa4c7 0xa5c7> \
            <0xa4c8 0xa5c8> <0xa4c9 0xa5c9> <0xa4ca 0xa5ca> \
            <0xa4cb 0xa5cb> <0xa4cc 0xa5cc> <0xa4cd 0xa5cd> \
            <0xa4ce 0xa5ce> <0xa4cf 0xa5cf> <0xa4d0 0xa5d0> \
            <0xa4d1 0xa5d1> <0xa4d2 0xa5d2> <0xa4d3 0xa5d3> \
            <0xa4d4 0xa5d4> <0xa4d5 0xa5d5> <0xa4d6 0xa5d6> \
            <0xa4d7 0xa5d7> <0xa4d8 0xa5d8> <0xa4d9 0xa5d9> \
            <0xa4da 0xa5da> <0xa4db 0xa5db> <0xa4dc 0xa5dc> \
            <0xa4dd 0xa5dd> <0xa4de 0xa5de> <0xa4df 0xa5df> \
```

```
            <0xa4e0 0xa5e0> <0xa4e1 0xa5e1> <0xa4e2 0xa5e2> \
            <0xa4e3 0xa5e3> <0xa4e4 0xa5e4> <0xa4e5 0xa5e5> \
            <0xa4e6 0xa5e6> <0xa4e7 0xa5e7> <0xa4e8 0xa5e8> \
            <0xa4e9 0xa5e9> <0xa4ea 0xa5ea> <0xa4eb 0xa5eb> \
            <0xa4ec 0xa5ec> <0xa4ed 0xa5ed> <0xa4ee 0xa5ee> \
            <0xa4ef 0xa5ef> <0xa4f0 0xa5f0> <0xa4f1 0xa5f1> \
            <0xa4f2 0xa5f2> <0xa4f3 0xa5f3>
#
# specification for supplementary code-set 2
#
LC_CTYPE2
iswchar6    0xa1 - 0xdf
iswchar14   0xa1 - 0xdf
```

The following is an example of an input file used to create the Korean character classification and conversion definition tables and store them in files designated as LC_CTYPE and LC_NUMERIC, respectively.

```
#################################################################
#                                                 #
# LC_CTYPE & LC_NUMERIC Information for Korean locale          #
#                                                 #
#################################################################

LC_CTYPE  LC_CTYPE
#
# ASCII Set Definition
#
isupper            0x41 - 0x5a

islower            0x61 - 0x7a

isdigit            0x30 - 0x39

isspace            0x20   0x9 - 0xd

ispunct            0x21 - 0x2f  0x3a - 0x40  0x5b - 0x60  0x7b - 0x7e

iscntrl            0x0 - 0x1f  0x7f

isblank            0x20

isxdigit     0x30 - 0x39  0x61 - 0x66  0x41 - 0x46

ul           <0x41 0x61>  <0x42 0x62>  <0x43 0x63>  <0x44 0x64> \
             <0x45 0x65>  <0x46 0x66>  <0x47 0x67>  <0x48 0x68> \
             <0x49 0x69>  <0x4a 0x6a>  <0x4b 0x6b>  <0x4c 0x6c> \
             <0x4d 0x6d>  <0x4e 0x6e>  <0x4f 0x6f>  <0x50 0x70> \
             <0x51 0x71>  <0x52 0x72>  <0x53 0x73>  <0x54 0x74> \
             <0x55 0x75>  <0x56 0x76>  <0x57 0x77>  <0x58 0x78> \
             <0x59 0x79>  <0x5a 0x7a>

cswidth            2:2,0:0,0:0


LC_NUMERIC  LC_NUMERIC
#
# Numeric Editing Information
#
decimal_point      .
thousands_sep      ,


LC_CTYPE1
#
# KSC5601 Set Definition
#
```

```
# Roman, Greek, Latin and Russian upper characters
isupper           0xa3c1 - 0xa3da  0xa5c1 - 0xa5d8 \
            0xa8a1 - 0xa8a4  0xa8a6  0xa8a8 - 0xa8af \
            0xaca1 - 0xacc1

# Roman, Greek, Latin and Russian lower characters
islower           0xa3e1 - 0xa3fa  0xa5e1 - 0xa5f8 \
            0xa9a1 - 0xa9b0 \
            0xacd1 - 0xacf1

isspace           0xa1a1

# Korean consonants and vowels
# old Korean consonants and vowels
# hangul
isphonogram 0xa4a1 - 0xa4d3  0xa4d5 - 0xa4fe \
            0xb0a1 - 0xb0fe  0xb1a1 - 0xb1fe  0xb2a1 - 0xb2fe \
            0xb3a1 - 0xb3fe  0xb4a1 - 0xb4fe  0xb5a1 - 0xb5fe \
            0xb6a1 - 0xb6fe  0xb7a1 - 0xb7fe  0xb8a1 - 0xb8fe \
            0xb9a1 - 0xb9fe  0xbaa1 - 0xbafe  0xbba1 - 0xbbfe \
            0xbca1 - 0xbcfe  0xbda1 - 0xbdfe  0xbea1 - 0xbefe \
            0xbfa1 - 0xbffe  0xc0a1 - 0xc0fe  0xc1a1 - 0xc1fe \
            0xc2a1 - 0xc2fe  0xc3a1 - 0xc3fe  0xc4a1 - 0xc4fe \
            0xc5a1 - 0xc5fe  0xc6a1 - 0xc6fe  0xc7a1 - 0xc7fe \
            0xc8a1 - 0xc8fe

# hanja
isideogram  0xcaa1 - 0xcafe  0xcba1 - 0xcbfe  0xcca1 - 0xccfe \
            0xcda1 - 0xcdfe  0xcea1 - 0xcefe  0xcfa1 - 0xcffe \
            0xd0a1 - 0xd0fe  0xd1a1 - 0xd1fe  0xd2a1 - 0xd2fe \
            0xd3a1 - 0xd3fe  0xd4a1 - 0xd4fe  0xd5a1 - 0xd5fe \
            0xd6a1 - 0xd6fe  0xd7a1 - 0xd7fe  0xd8a1 - 0xd8fe \
            0xd9a1 - 0xd9fe  0xdaa1 - 0xdafe  0xdba1 - 0xdbfe \
            0xdca1 - 0xdcfe  0xdda1 - 0xddfe  0xdea1 - 0xdefe \
            0xdfa1 - 0xdffe  0xe0a1 - 0xe0fe  0xe1a1 - 0xe1fe \
            0xe2a1 - 0xe2fe  0xe3a1 - 0xe3fe  0xe4a1 - 0xe4fe \
            0xe5a1 - 0xe5fe  0xe6a1 - 0xe6fe  0xe7a1 - 0xe7fe \
            0xe8a1 - 0xe8fe  0xe9a1 - 0xe9fe  0xeaa1 - 0xeafe \
            0xeba1 - 0xebfe  0xeca1 - 0xecfe  0xeda1 - 0xedfe \
            0xeea1 - 0xeefe  0xefa1 - 0xeffe  0xf0a1 - 0xf0fe \
            0xf1a1 - 0xf1fe  0xf2a1 - 0xf2fe  0xf3a1 - 0xf3fe \
            0xf4a1 - 0xf4fe  0xf5a1 - 0xf5fe  0xf6a1 - 0xf6fe \
            0xf7a1 - 0xf7fe  0xf8a1 - 0xf8fe  0xf9a1 - 0xf9fe \
            0xfaa1 - 0xfafe  0xfba1 - 0xfbfe  0xfca1 - 0xfcfe \
            0xfda1 - 0xfdfe

isenglish   0xa3c1 - 0xa3da  0xa3e1 - 0xa3fa

isnumber    0xa3b0 - 0xa3b9

# other printable characters
iswchar6    0xa1a2 - 0xa1fe  0xa2a1 - 0xa2e5  0xa3a1 - 0xa3af \
            0xa3ba - 0xa3c0  0xa3db - 0xa3e0  0xa3fb - 0xa3fe \
            0xa5a1 - 0xa5aa  0xa5b0 - 0xa5b9  0xa6a1 - 0xa6e4 \
            0xa7a1 - 0xa7ef  0xa8b1 - 0xa8fe  0xa9b1 - 0xa9fe \
```

```
            0xaaa1 - 0xaaf3  0xaba1 - 0xabf6

# KSC5601 whole characters
iswchar9    0xa1a1 - 0xa1fe  0xa2a1 - 0xa2e5  0xa3a1 - 0xa3fe \
            0xa4a1 - 0xa4fe  0xa5a1 - 0xa5aa  0xa5b0 - 0xa5b9 \
            0xa5c1 - 0xa5d8  0xa5e1 - 0xa5f8  0xa6a1 - 0xa6e4 \
            0xa7a1 - 0xa7ef  0xa8a1 - 0xa8a4  0xa8a6 \
            0xa8a8 - 0xa8af  0xa8b1 - 0xa8fe  0xa9a1 - 0xa9fe \
            0xaaa1 - 0xaaf3  0xaba1 - 0xabf6  0xaca1 - 0xacc1 \
            0xacd1 - 0xacf1  0xb0a1 - 0xb0fe  0xb1a1 - 0xb1fe \
            0xb2a1 - 0xb2fe  0xb3a1 - 0xb3fe  0xb4a1 - 0xb4fe \
            0xb5a1 - 0xb5fe  0xb6a1 - 0xb6fe  0xb7a1 - 0xb7fe \
            0xb8a1 - 0xb8fe  0xb9a1 - 0xb9fe  0xbaa1 - 0xbafe \
            0xbba1 - 0xbbfe  0xbca1 - 0xbcfe  0xbda1 - 0xbdfe \
            0xbea1 - 0xbefe  0xbfa1 - 0xbffe  0xc0a1 - 0xc0fe \
            0xc1a1 - 0xc1fe  0xc2a1 - 0xc2fe  0xc3a1 - 0xc3fe \
            0xc4a1 - 0xc4fe  0xc5a1 - 0xc5fe  0xc6a1 - 0xc6fe \
            0xc7a1 - 0xc7fe  0xc8a1 - 0xc8fe  0xcaa1 - 0xcafe \
            0xcba1 - 0xcbfe  0xcca1 - 0xccfe  0xcda1 - 0xcdfe \
            0xcea1 - 0xcefe  0xcfa1 - 0xcffe  0xd0a1 - 0xd0fe \
            0xd1a1 - 0xd1fe  0xd2a1 - 0xd2fe  0xd3a1 - 0xd3fe \
            0xd4a1 - 0xd4fe  0xd5a1 - 0xd5fe  0xd6a1 - 0xd6fe \
            0xd7a1 - 0xd7fe  0xd8a1 - 0xd8fe  0xd9a1 - 0xd9fe \
            0xdaa1 - 0xdafe  0xdba1 - 0xdbfe  0xdca1 - 0xdcfe \
            0xdda1 - 0xddfe  0xdea1 - 0xdefe  0xdfa1 - 0xdffe \
            0xe0a1 - 0xe0fe  0xe1a1 - 0xe1fe  0xe2a1 - 0xe2fe \
            0xe3a1 - 0xe3fe  0xe4a1 - 0xe4fe  0xe5a1 - 0xe5fe \
            0xe6a1 - 0xe6fe  0xe7a1 - 0xe7fe  0xe8a1 - 0xe8fe \
            0xe9a1 - 0xe9fe  0xeaa1 - 0xeafe  0xeba1 - 0xebfe \
            0xeca1 - 0xecfe  0xeda1 - 0xedfe  0xeea1 - 0xeefe \
            0xefa1 - 0xeffe  0xf0a1 - 0xf0fe  0xf1a1 - 0xf1fe \
            0xf2a1 - 0xf2fe  0xf3a1 - 0xf3fe  0xf4a1 - 0xf4fe \
            0xf5a1 - 0xf5fe  0xf6a1 - 0xf6fe  0xf7a1 - 0xf7fe \
            0xf8a1 - 0xf8fe  0xf9a1 - 0xf9fe  0xfaa1 - 0xfafe \
            0xfba1 - 0xfbfe  0xfca1 - 0xfcfe  0xfda1 - 0xfdfe

ul          <0xa3c1 0xa3e1>  <0xa3c2 0xa3e2>  <0xa3c3 0xa3e3> \
            <0xa3c4 0xa3e4>  <0xa3c5 0xa3e5>  <0xa3c6 0xa3e6> \
            <0xa3c7 0xa3e7>  <0xa3c8 0xa3e8>  <0xa3c9 0xa3e9> \
            <0xa3ca 0xa3ea>  <0xa3cb 0xa3eb>  <0xa3cc 0xa3ec> \
            <0xa3cd 0xa3ed>  <0xa3ce 0xa3ee>  <0xa3cf 0xa3ef> \
            <0xa3d0 0xa3f0>  <0xa3d1 0xa3f1>  <0xa3d2 0xa3f2> \
            <0xa3d3 0xa3f3>  <0xa3d4 0xa3f4>  <0xa3d5 0xa3f5> \
            <0xa3d6 0xa3f3>  <0xa3d7 0xa3f7>  <0xa3d8 0xa3f8>
            <0xa3d9 0xa3f9>  <0xa3da 0xa3fa>  \
            <0xa5c1 0xa5e1>  <0xa5c2 0xa5e2>  <0xa5c3 0xa5e3> \
            <0xa5c4 0xa5e4>  <0xa5c5 0xa5e5>  <0xa5c6 0xa5e3> \
            <0xa5c7 0xa5e7>  <0xa5c8 0xa5e8>  <0xa5c9 0xa5e9> \
            <0xa5ca 0xa5ea>  <0xa5cb 0xa5eb>  <0xa5cc 0xa5ec> \
            <0xa5cd 0xa5ed>  <0xa5ce 0xa5ee>  <0xa5cf 0xa5ef> \
            <0xa5d0 0xa5f0>  <0xa5d1 0xa5f1>  <0xa5d2 0xa5f2> \
            <0xa5d3 0xa5f3>  <0xa5d4 0xa5f4>  <0xa5d5 0xa5f5> \
            <0xa5d6 0xa5f3>  <0xa5d7 0xa5f7>  <0xa5d8 0xa5f8> \
            <0xa8a1 0xa9a1>  <0xa8a2 0xa9a2>  <0xa8a3 0xa9a3> \
            <0xa8a4 0xa9a4>  <0xa8a6 0xa9a6>  <0xa8a8 0xa9a8> \
```

```
<0xa8a9 0xa9a9>  <0xa8aa 0xa9aa>  <0xa8ab 0xa9ab> \
<0xa8ac 0xa9ac>  <0xa8ad 0xa9ad>  <0xa8ae 0xa9ae> \
<0xa8af 0xa9af> \
<0xaca1 0xacd1>  <0xaca2 0xacd2>  <0xaca3 0xacd3> \
<0xaca4 0xacd4>  <0xaca5 0xacd5>  <0xaca6 0xacd6> \
<0xaca7 0xacd7>  <0xaca8 0xacd8>  <0xaca9 0xacd9> \
<0xacaa 0xacda>  <0xacab 0xacdb>  <0xacac 0xacdc> \
<0xacad 0xacdd>  <0xacae 0xacde>  <0xacaf 0xacdf> \
<0xacb0 0xace0>  <0xacb1 0xace1>  <0xacb2 0xace2> \
<0xacb3 0xace3>  <0xacb4 0xace4>  <0xacb5 0xace5> \
<0xacb6 0xace6>  <0xacb7 0xace7>  <0xacb8 0xace8> \
<0xacb9 0xace9>  <0xacba 0xacea>  <0xacbb 0xaceb> \
<0xacbc 0xacec>  <0xacbd 0xaced>  <0xacbe 0xacee> \
<0xacbf 0xacef>  <0xacc0 0xacf0>  <0xacc1 0xacf1>
```

Appendix E: xctype.h Example

This is an excerpt from an example of the xctype.h header file for the Japanese language:

```
/*      xctype.h for Japanese Language                */
/*    definition for native language function         */

#define isjis(c)  _iswctype(c,_E9)  /* JIS X0208 all characters */
#define isjspecial(c)    isspecial(c)          /* JIS X0208 special char */
#define isjparen(c)     _iswctype(c,_E10)      /* JIS X0208 parenthesis */
#define isjdigit(c)      isnumber(c)           /* JIS X0208 digit */
#define isjalpha(c)      isenglish(c)          /* JIS X0208 English char */
#define isjhira(c)      _iswctype(c,_E11)      /* JIS X0208 hiragana */
#define isjkata(c)      _iswctype(c,_E12)      /* JIS X0208 katakana */
#define isjkanji(c)      isideogram(c)         /* JIS X0208 kanji */
#define isjhankana(c)   _iswctype(c,_E14)      /* JIS X0201 katakana */
#define tojhira(c)      _trwctype(c,_E12)      /* JIS X0208 katakana to hiragana */
#define tojkata(c)      _trwctype(c,_E11)      /* JIS X0208 hiragana to katakana */
```

Appendix E: xctype.h Example

This is an excerpt from an example of the xctype.h header file for the Korean language:

```
/* xctype.h for Korean locale */

#defineishangul(c)   isphonogram(c)       /* KS C5601 hangul characters */
#defineishanja(c)    isideogram(c)        /* KS C5601 hanja characters */
#define isksc(c)     _iswctype(c, _E9)    /* KS C5601 all characters */
```

Appendix F: LC_TIME Examples

The following are two examples of time formatting information to be stored in a file LC_TIME.

Example for U.S.

```
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
January
February
March
April
May
June
July
August
September
October
November
December
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
%H:%M:%S
```

```
%m/%d/%y
%a %b %d %H:%M:%S %Y
AM
PM
%a %b %e %T %Z %Y
```

Example for French language.

```
jan
fév
mar
avr
mai
jun
jui
aoû
sep
oct
nov
déc
janvier
février
mars
avril
mai
juin
juillet
août
septembre
octobre
novembre
décembre
dim
lun
mar
mer
jeu
ven
sam
dimanche
lundi
mardi
mercredi
jeudi
```

```
vendredi
samedi
%H:%M:%S
%d/%m/%y
%A %d %B %Y %H:%M:%S %Z
AM
PM
%a %d %b %Y %T %Z
```

Appendix G: LC_COLLATE Example

The following example shows the collation specification required to support a hypothetical telephone book sorting sequence.

The sorting sequence is defined by the following rules:

1. Upper and lower case letters must be sorted together, but upper case letters have precedence over lower case letters.
2. All special characters and punctuation should be ignored.
3. Digits must be sorted as their alphabetic counterparts (e.g., 0 as zero, 1 as one).
4. The Ch, ch, CH combinations must be collated between C and D.
5. V and W, v and w must be collated together.

The input specification file to colltbl will contain:

```
code-set   telephone

order is   (A;a);(B;b);(C;c);(CH;Ch;ch);(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;W};{v;w};(X;x);(Y;y);(Z;z)

substitute "0" with "zero"
substitute "1" with "one"
substitute "2" with "two"
substitute "3" with "three"
substitute "4" with "four"
substitute "5" with "five"
substitute "6" with "six"
substitute "7" with "seven"
substitute "8" with "eight"
substitute "9" with "nine"
```

Appendix H: LC_MONETARY Examples

The following are examples of input files used to to create the Italy, Netherlands, Norway, and Switzerland(certainregions) Taiwan,R.O.C. Japan Korea monetary database definition files. Each of these will be stored in a locale-specific LC_MONETARY file.

The monetary formats are:

Country	Positive format	Negative format	International format
Italy	L.1.234	-L.1234	ITL.1.234
Netherlands	F 1.234,56	F -1.234,56	NLG 1.234,56
Norway	kr1.234,56	kr1.234,56-	NOK 1.234,56
Switzerland	SFrs.1,234.56	SFrs.1,234.56C	CHF 1,234.56
Taiwan, R.O.C.	NT$1,234.56	-NT$1,234.56	TWD 1,234.56

H.1 Italy Example

```
int_curr_symbol     "ITL."
currency_symbol     "L."
mon_decimal_point   ""
mon_thousands_sep   "."
mon_grouping        "3"
positive_sign       ""
negative_sign       "-"
int_frac_digits     0
frac_digits         0
p_cs_precedes       1
p_sep_by_space      0
n_cs_precedes       1
n_sep_by_space      0
p_sign_posn         1
n_sign_posn         1
```

H.2 Netherlands Example

```
int_curr_symbol     "NLG "
currency_symbol     "F"
mon_decimal_point   ","
mon_thousands_sep   "."
mon_grouping        "3"
```

```
positive_sign       ""
negative_sign       "-"
int_frac_digits     2
frac_digits         2
p_cs_precedes       1
p_sep_by_space      1
n_cs_precedes       1
n_sep_by_space      1
p_sign_posn         1
n_sign_posn         4
```

H.3 Norway Example

```
int_curr_symbol     "NOK"
currency_symbol     "kr"
mon_decimal_point   ","
mon_thousands_sep   "."
mon_grouping        "3"
positive_sign       ""
negative_sign       "-"
int_frac_digits     2
frac_digits         2
p_cs_precedes       1
p_sep_by_space      0
n_cs_precedes       1
n_sep_by_space      0
p_sign_posn         1
n_sign_posn         2
```

H.4 Example For Switzerland (certain regions)

```
int_curr_symbol     "CHF "
currency_symbol     "SFrs"
mon_decimal_point   "."
mon_thousands_sep   ","
mon_grouping        "3"
positive_sign       ""
negative_sign       "C"
```

```
int_frac_digits      2
frac_digits          2
p_cs_precedes        1
p_sep_by_space       0
n_cs_precedes        1
n_sep_by_space       0
p_sign_posn          1
n_sign_posn          2
```

H.5 Example for Taiwan R.O.C.

```
int_curr_symbol      "TWD "
currency_symbol      "NT$"
mon_decimal_point    "."
mon_thousands_sep    ","
mon_grouping         "3"
positive_sign        ""
negative_sign        "-"
int_frac_digits      2
frac_digits          2
p_cs_precedes        1
p_sep_by_space       0
n_cs_precedes        1
n_sep_by_space       0
p_sign_posn          1
n_sign_posn          1
```

H.6 Example for Japan

```
int_curr_symbol      "JPY "
currency_symbol      "¥"
mon_decimal_point    ""
mon_thousands_sep    ","
mon_grouping         "3; "
positive_sign        ""
negative_sign        "-"
```

```
int_frac_digits     0
frac_digits         0
p_cs_precedes       1
p_sep_by_space      0
n_cs_precedes       1
n_sep_by_space      0
p_sign_posn         1
n_sign_posn         4
```

H.7 Example for Korea

```
int_curr_symbol     "KRW"
currency_symbol     "₩"
mon_decimal_point   "."
mon_thousands_sep   ","
mon_grouping        "3; "
positive_sign       ""
negative_sign       "-"
int_frac_digits     2
frac_digits         2
p_cs_precedes       1
p_sep_by_space      0
n_cs_precedes       1
n_sep_by_space      0
p_sign_posn         4
n_sign_posn         4
```

Appendix I: kbdcomp Examples

kbdcomp compiles tables for use with the kbd STREAMS module. Source files for kbdcomp are a series of table declarations. The following are examples of such tables.

The first example is an input file to *kbdcomp* to create a keyboard mapping table for ISO 8859-1:

```
# A complete sample 8859-1 map, using dead keys.
#
#       Compose key = #
#       The following keys are dead:
#               `       grave
#               ´       acute
#               "       umlaut
#               ^       circumflex
#                       tilde
#
# The compose and dead keys can be entered by doubling
# them, e.g. ## for #
#
map full (8859-1.dk) {
        define(comp "#")  # compose key
        comp(´#´ ´#´)           # compose x 2

        define(gr ´`´)          # accent grave
        gr(´`´ ´`´)       # backquote

        define(ac ´\047´) # accent acute
        ac(´\047´ ´\047´) # single quote

        define(um ´"´)          # umlaut
        um(´"´ ´"´)       # double quote

        define(cir ´^´)         # circumflex
        cir(´^´ ´^´)            # carat

        define(til ´ ´)         # tilde
        til(´ ´ ´ ´)            # tilde alone

        comp(´!´ ´\241´)  # Spanish initial "!"
        comp("c/" ´\242´) # US cents
        comp(´L´ ´\243´)  # British Pound
        comp("xo" ´\244´) # currency
        comp(´Y´ ´\245´)  # Yen
        comp(´|´ ´\246´)  # broken bar
        comp(´S´ ´\247´)  # section sign
        comp(´"´ ´\250´)  # umlaut alone
```

```
comp("co" ´\251´) # copyright
comp("^a" ´\252´) # a sup (fem. ordinal)
comp(´<´ ´\253´)  # <<
comp(´-´ ´\254´)  # logical not
                  # soft hyphen (255) omitted
comp("rg" ´\256´) # registered trademark
comp("_" ´\257´)  # macron
comp("dg" ´\260´) # degrees
comp("+-" ´\261´) # plus or minus
comp("^2" ´\262´) # 2 sup
comp("^3" ´\263´) # 3 sup
comp("´" ´\264´)  # accent acute
comp("mu" ´\265´) # micro sign
comp(´P´ ´\266´)  # Pilcrow (paragraph)
comp(´.´ ´\267´)  # middle dot
comp(´,´ ´\270´)  # cedilla
comp("^1" ´\271´) # 1 sup
comp("^o" ´\272´) # o sup (masc. ordinal)
comp(´>´ ´\273´)  # >>
comp("/14" ´\274´)      # 1/4 frac.
comp("/12" ´\275´)      # 1/2 frac.
comp("/34" ´\276´)      # 3/4 frac.
comp(´?´ ´\277´)  # Spanish initial "?"

gr(A ´\300´)
ac(A ´\301´)
cir(A ´\302´)
til(A ´\303´)
um(A ´\304´)
comp(Ao ´\305´)   # A with ring above
comp(AE ´\306´)   # AE ligature
comp("C," ´\307´) # C cedilla
gr(E ´\310´)
ac(E ´\311´)
cir(E ´\312´)
um(E ´\313´)
gr(I ´\314´)
ac(I ´\315´)
cir(I ´\316´)
um(I ´\317´)

comp("Eth" ´\320´)      # Capital Eth
til(N ´\321´)
gr(O ´\322´)
ac(O ´\323´)
cir(O ´\324´)
```

```
        til(O ´\325´)
        um(O ´\326´)
        comp("xx" ´\327´) # multiplication
        comp("O/" ´\330´) # O w/ oblique stroke
        gr(U ´\331´)
        ac(U ´\332´)
        cir(U ´\333´)
        um(U ´\334´)
        gr(Y ´\335´)
        comp(Th ´\336´)   # Capital Thorn
        comp(ss ´\337´)   # German sharp S
        gr(a ´\340´)
        ac(a ´\341´)
        cir(a ´\342´)
        til(a ´\343´)
        um(a ´\344´)
        comp(ao ´\345´)   # a with ring above
        comp(ae ´\346´)   # ae ligature
        comp("c," ´\347´) # c cedilla
        gr(e ´\350´)
        ac(e ´\351´)
        cir(e ´\352´)
        um(e ´\353´)
        gr(i ´\354´)
        ac(i ´\355´)
        cir(i ´\356´)
        um(i ´\357´)
        comp("eth" ´\360´)      # small eth
        til(n ´\361´)
        gr(o ´\362´)
        ac(o ´\363´)
        cir(o ´\364´)
        til(o ´\365´)
        um(o ´\366´)
        comp(":-" ´\367´) # division
        comp("/o" ´\370´)
        gr(u ´\371´)
        ac(u ´\372´)
        cir(u ´\373´)
        um(u ´\374´)
        ac(y ´\375´)
        comp("th" ´\376´)
        um(y ´\377´)
}
```

The second example is an input file to *kbdcomp* to create three keyboard mapping tables for English conversion:

```
#
# Sample ASCII upper-to-lower, lower-to-upper, and case-swap maps.
#
map (uclc) {
      keylist("ABCDEFGHIJKLMNOPQRSTUVWXYZ"
            "abcdefghijklmnopqrstuvwxyz")
}

map (lcuc) {
      keylist("abcdefghijklmnopqrstuvwxyz"
            "ABCDEFGHIJKLMNOPQRSTUVWXYZ")

}
map (uclc_lcuc) {
      keylist("ABCDEFGHIJKLMNOPQRSTUVWXYZ"
            "abcdefghijklmnopqrstuvwxyz")
      keylist("abcdefghijklmnopqrstuvwxyz"
            "ABCDEFGHIJKLMNOPQRSTUVWXYZ")
}
```

The third example is an input file to *kbdcomp* to create a keyboard mapping table for Dvorak keyboard mapping.

```
#
# This is a sample Dvorak keyboard arrangement.  It may not be
# complete and is not guaranteed, but the alphabet is hopefully
# in the right place.
#
map (Dvorak) {
      keylist("qwertyuiop" ";,.pyfgcrl")
      keylist("QWERTYUIOP" ":/?PYFGCRL")
      keylist("asdfghjkl;" "aoeuidhtns")
      keylist("ASDFGHJKL:" "AOEUIDHTNS")
      keylist("zxcvbnm,./" "<qjkxbmwvz")
      keylist("ZXCVBNM<>?" ">QJKXBMWVZ")
}
```

The fourth example is a portion of a mapping table which can be used for conversion from MS-Kanji (Shift-JIS) to Japanese EUC.

```
# Since the full table contains more than 6,000 lines,
# displayed here is a table for only first 100 characters
# of JIS X0208 Kanji (in codeset 1).
```

```
map full (stou) {
      string("\201@" "\241\241")
      string("\201A" "\241\242")
      string("\201B" "\241\243")
      string("\201C" "\241\244")
      string("\201D" "\241\245")
      string("\201E" "\241\246")
      string("\201F" "\241\247")
      string("\201G" "\241\250")
      string("\201H" "\241\251")
      string("\201I" "\241\252")
      string("\201J" "\241\253")
      string("\201K" "\241\254")
      string("\201L" "\241\255")
      string("\201M" "\241\256")
      string("\201N" "\241\257")
      string("\201O" "\241\260")
      string("\201P" "\241\261")
      string("\201Q" "\241\262")
      string("\201R" "\241\263")
      string("\201S" "\241\264")
      string("\201T" "\241\265")
      string("\201U" "\241\266")
      string("\201V" "\241\267")
      string("\201W" "\241\270")
      string("\201X" "\241\271")
      string("\201Y" "\241\272")
      string("\201Z" "\241\273")
      string("\201[" "\241\274")
      string("\201\\" "\241\275")
      string("\201]" "\241\276")
      string("\201^" "\241\277")
      string("\201_" "\241\300")
      string("\201`" "\241\301")
      string("\201a" "\241\302")
      string("\201b" "\241\303")
      string("\201c" "\241\304")
      string("\201d" "\241\305")
      string("\201e" "\241\306")
      string("\201f" "\241\307")
      string("\201g" "\241\310")
      string("\201h" "\241\311")
      string("\201i" "\241\312")
      string("\201j" "\241\313")
      string("\201k" "\241\314")
      string("\201l" "\241\315")
```

```
string("\201m" "\241\316")
string("\201n" "\241\317")
string("\201o" "\241\320")
string("\201p" "\241\321")
string("\201q" "\241\322")
string("\201r" "\241\323")
string("\201s" "\241\324")
string("\201t" "\241\325")
string("\201u" "\241\326")
string("\201v" "\241\327")
string("\201w" "\241\330")
string("\201x" "\241\331")
string("\201y" "\241\332")
string("\201z" "\241\333")
string("\201{" "\241\334")
string("\201|" "\241\335")
string("\201}" "\241\336")
string("\201 " "\241\337")
string("\201\200" "\241\340")
string("\201\201" "\241\341")
string("\201\202" "\241\342")
string("\201\203" "\241\343")
string("\201\204" "\241\344")
string("\201\205" "\241\345")
string("\201\206" "\241\346")
string("\201\207" "\241\347")
string("\201\210" "\241\350")
string("\201\211" "\241\351")
string("\201\212" "\241\352")
string("\201\213" "\241\353")
string("\201\214" "\241\354")
string("\201\215" "\241\355")
string("\201\216" "\241\356")
string("\201\217" "\241\357")
string("\201\220" "\241\360")
string("\201\221" "\241\361")
string("\201\222" "\241\362")
string("\201\223" "\241\363")
string("\201\224" "\241\364")
string("\201\225" "\241\365")
string("\201\226" "\241\366")
string("\201\227" "\241\367")
string("\201\230" "\241\370")
string("\201\231" "\241\371")
string("\201\232" "\241\372")
string("\201\233" "\241\373")
```

```
        string("\201\234" "\241\374")
        string("\201\235" "\241\375")
        string("\201\236" "\241\376")
        string("\201\237" "\242\241")
        string("\201\240" "\242\242")
        string("\201\241" "\242\243")
        string("\201\242" "\242\244")
        string("\201\243" "\242\245")
        string("\201\244" "\242\246")
}
```

The last example is an input file to create a Korean keyboard mapping table.

```
#
# This is an input file for kbdcomp to create a Korean keyboard mapping table
# for KSC5601.1987-0.
#

map full (stou) {
        string("a" "\261")
        string("b" "\320")
        string("c" "\272")
        string("d" "\267")
        string("e" "\247")
        string("f" "\251")
        string("g" "\276")
        string("h" "\307")
        string("i" "\301")
        string("j" "\303")
        string("k" "\277")
        string("l" "\323")
        string("m" "\321")
        string("n" "\314")
        string("o" "\300")
        string("p" "\304")
        string("q" "\262")
        string("r" "\241")
        string("s" "\244")
        string("t" "\265")
        string("u" "\305")
        string("v" "\275")
        string("w" "\270")
        string("x" "\274")
        string("y" "\313")
        string("z" "\273")
        string("Q" "\263")
        string("W" "\271")
        string("E" "\250")
```

```
        string("R" "\242")
        string("T" "\266")
        string("O" "\302")
        string("P" "\306")
}
```

Appendix J: iconv Example

The following is an example of an input file used by *kbdcomp*(1M) to create the **646fr** to **8859-1** code-set conversion table.

```
# From code-set:  French ascii
# To code-set:    European 8859-1
# Based on: 8859.646
# Notes:

map full (646fr.8859) {

      string(\043 \243) # hash -> pound
      string(\100 \340) # @ -> a grave
      string('[' \260)  # -> ring above
      string(\134 \347) # -> cedilla
      string(']' \247)  # -> paragraph sign
      string('{' \351)  # -> e acute
      string('|' \371)  # -> u grave
      string('}' \350)  # -> e grave
      string(' ' \250)  # -> umlaut

      string(\200 '_')
      string(\201 '_')
      string(\202 '_')
      string(\203 '_')
      string(\204 '_')
      string(\205 '_')
      string(\206 '_')
      string(\207 '_')

      string(\210 '_')
      string(\211 '_')
      string(\212 '_')
      string(\213 '_')
      string(\214 '_')
      string(\215 '_')
      string(\216 '_')
      string(\217 '_')

      string(\220 '_')
      string(\221 '_')
      string(\222 '_')
      string(\223 '_')
      string(\224 '_')
      string(\225 '_')
      string(\226 '_')
      string(\227 '_')
```

```
string(\230 '_')
string(\231 '_')
string(\232 '_')
string(\233 '_')
string(\234 '_')
string(\235 '_')
string(\236 '_')
string(\237 '_')

string(\240 '_')
string(\241 '_')
string(\242 '_')
string(\243 '_')
string(\244 '_')
string(\245 '_')
string(\246 '_')
string(\247 '_')

string(\250 '_')
string(\251 '_')
string(\252 '_')
string(\253 '_')
string(\254 '_')
string(\255 '_')
string(\256 '_')
string(\257 '_')

string(\260 '_')
string(\261 '_')
string(\262 '_')
string(\263 '_')
string(\264 '_')
string(\265 '_')
string(\266 '_')
string(\267 '_')

string(\270 '_')
string(\271 '_')
string(\272 '_')
string(\273 '_')
string(\274 '_')
string(\275 '_')
string(\276 '_')
string(\277 '_')

string(\300 '_')
```

```
string(\301 '_')
string(\302 '_')
string(\303 '_')
string(\304 '_')
string(\305 '_')
string(\306 '_')
string(\307 '_')

string(\310 '_')
string(\311 '_')
string(\312 '_')
string(\313 '_')
string(\314 '_')
string(\315 '_')
string(\316 '_')
string(\317 '_')

string(\320 '_')
string(\321 '_')
string(\322 '_')
string(\323 '_')
string(\324 '_')
string(\325 '_')
string(\326 '_')
string(\327 '_')

string(\330 '_')
string(\331 '_')
string(\332 '_')
string(\333 '_')
string(\334 '_')
string(\335 '_')
string(\336 '_')
string(\337 '_')

string(\340 '_')
string(\341 '_')
string(\342 '_')
string(\343 '_')
string(\344 '_')
string(\345 '_')
string(\346 '_')
string(\347 '_')

string(\350 '_')
string(\351 '_')
```

```
        string(\352 '_')
        string(\353 '_')
        string(\354 '_')
        string(\355 '_')
        string(\356 '_')
        string(\357 '_')

        string(\360 '_')
        string(\361 '_')
        string(\362 '_')
        string(\363 '_')
        string(\364 '_')
        string(\365 '_')
        string(\366 '_')
        string(\367 '_')

        string(\370 '_')
        string(\371 '_')
        string(\372 '_')
        string(\373 '_')
        string(\374 '_')
        string(\375 '_')
        string(\376 '_')
        string(\377 '_')
}
```

The output of *kbdcomp* should be installed in /usr/lib/iconv/646fr.8859.t

The configuration file One line must be added to the file /usr/lib/iconv/iconv_data to install this code-set conversion table.

```
646fr 8859  646fr.8859  646fr.8859.t
```

Note: Refer to the *European Language Supplement 4.0 for UNIX System V Release 4.0*, for more information.

Appendix K: International Functions & Commands

System V includes the following functions to provide international capabilities and information in an application.

Internationalization Functions or Commands	Uninternationalized Counterparts	Description
catopen(3C)		Open a message catalog
catgets(3C)		Read a message from a message catalog
catclose(3C)		Close a message catalog
colltbl(1)		Create collation database
ecvt(3C)		Convert floating-point number to string
exstr(1)		Extract strings from source files
gencat(1)		Generate a formatted message catalog
getdate(3C)		Convert user format date and time
gettxt(3C),gettxt(1)		Retrieve a text string from a message data base
getwc,getwchar,fgetwc(3W)	getc,getchar,fgetc(3C)	get a character from a stream
getwidth(3W)		Get supplementary code-set information
getws,fgetws(3W)	gets,fgets(3C)	Get a string from a stream
iconv(1)		Code-set conversion utility
kbdcomp(1), kbdload(1)		Compile, load, kbd tables
kbdpipe(1)		Use the kbd module in a pipeline
kbdpipe(1), kbdset(1)		attach kbd mapping tables, set mode
localeconv(3C)		Get numeric formatting information
mbtowc,wctomb,mblen(3C)		Multibyte character conversion
mbstowcs,wctombs(3C)		Multibyte string conversion
mkmsgs(1)		Create message files for use by gettxt
montbl(1)		Create monetary database
strerror(3C)		Get error message string
printf,fprintf,sprintf(3W)	printf,fprintf,sprintf(3C)	Print formatted output
putwc,putwchar,fputwc(3W)	putc,putchar,fputc(3C)	
putws,fputws(3W)	puts,fputs(3C)	Put a character string on a stream
scanf,fscanf,sscanf(3W)	scanf,fscanf,sscanf(3C)	Convert formatted input
setlocale(3C)		Set, change or query the locale of a process
srchtxt(1)		Search for a text string
strftime(3C)	ctime(3C)	
strcoll(3C)		String collation
strxfrm(3C)		String transformation
ungetwc(3W)	ungetc(3C)	
wchrtbl(1)	chrtbl(1)	Generate character classification/conversion and numeric tables
wconv(3W)		Translate characters
wctype(3W)	ctype(3C)	Classify ASCII and supplementary code-set characters

Internationalization Functions or Commands	Uninternationalized Counterparts	Description
widec(3W)		Multibyte character I/O routines
wctomb(3C)		Convert a wide character to a mulbtiye characte
wcstombs(3C)		Convert a wide character string to a mulbtiye character string
wstring(3W)		`wchar_t` string operations and type transformation

Refer to

- The *SVR4 MNLS Product Overview*,
- The sections titled "Multibyte Characters and Wide Characters" and "Internationalization" of the *SVR4 ANSI C Transition Guide*,
- The *SVR4 Programmer's Reference Manual*
- Section 4.2, titled "PROGRAM LOCALISATION", of the *X/Open Portability Guide - Volume 3 XSI Supplementary Definitions*

for more information.

PLEASE DISCARD THIS PAGE!!!

Table of Contents