Open Source Project for Unicode Locales

- Open Source Linux Locale Data in XML -

Kentaro Noji
Tetsuji Orita

Globalization Center of Competency
Yamato Software Laboratory, IBM Japan Ltd.

Abstract: We've developed over 140 Linux locales and made them available via IBM developersWorks under the IBM Public License-open source license. These locales have been developed based on the Unicode online database, collation keys and locale data in XML. The locale data in XML was generated from the ICU (Internationalization Class for Unicode). Java also uses the ICU architecture. Consequently the Linux locale sensitive functions are equivalent between ICU and Java. We will maintain all the locale data through these XML files as the master locale data. In this paper, we describe the overall scope of this project and the technical methodology used to develop Linux locales. We also introduce a tool to display this locale data in human readable format via the Web.

1. Introduction

Open source development is the most attractive software development methodology today. We developed Unicode locales in order to provide them to the Linux open source community. The Universal Locales for Linux is a package of Unicode locales that is based on the Unicode Standard V3.0 [1]. The Universal Locales for Linux was created using the UCA(Unicode Collation Algorithm) [2], Unicode data [3] in Unicode V3.0 and ICU–International Components for Unicode [4].

Linux is a UNIX clone operating system. It provides functionalities similar to UNIX as it is specified by POSIX [5] and XPG [6]. POSIX and XPG define an internationalization specification, but Linux follows the LI18NUX 2000 Globalization Specification V1.0 [7] that is specified by the Linux Internationalization Initiative [8]. Linux internationalization is mainly handled by the GlibC(GNU C library) [9]. The GlibC implements stable internationalization functionalities from V2.2 up, and the Universal Locale supports the GlibC V2.2. The Universal Locales provides over 140 Linux locales that constitute a superset of the locales that the LI18NUX 2000 Globalization specification defines, the locales that the ICU provides, and the existing Linux locales. The Universal Locales are available from IBM developerWorks [10].

The following is the objective of our locale project.

A) Define master locale data in XML and develop the transformer from the master locale data to the Linux locale data.
B) Use the open source development methodology for locale development and provide Unicode locales as open source resources.

In this paper, we describe the Locale development process in Chapter 2, the design of the locales in XML in Chapter 3, the implementation of the transformer from XML to a Linux locale in Chapter 4, and a locale data verification tool in Chapter 5.

2. Locale Development Process

2.1 The development cycle and the existing internationalization resources.

Linux locales have been developed in the open source community but the existing Linux locales are only suitable for encodings such as ISO 8859-1, eucJP, and so on. Because our functional objective is to follow both Unicode and ICU, we created all locales directly from the master locale data in XML. Fig. 1 shows the relation between existing resources such as ICU, Unicode, and our development process. The master locales were created from the ICU locales, and they are maintained as the master locale data. Using master locale data is effective in maintaining consistency among the ICU, the Universal Locales, and the XML locales. When a new locale is added, the locale data in XML is added, not platform specific locale data. Bug reports are collected from the open source community. The bug fix is applied to all locale data by fixing the master locale in XML. By virtue of this development cycle, the locale data can be verified by the native speakers from all over the world, and the locale data is more credible. The bug fixes are easily delivered as open source.

The XML file variant.xml in the master locales is the locale data for the GlibC V2.2 specific categories.

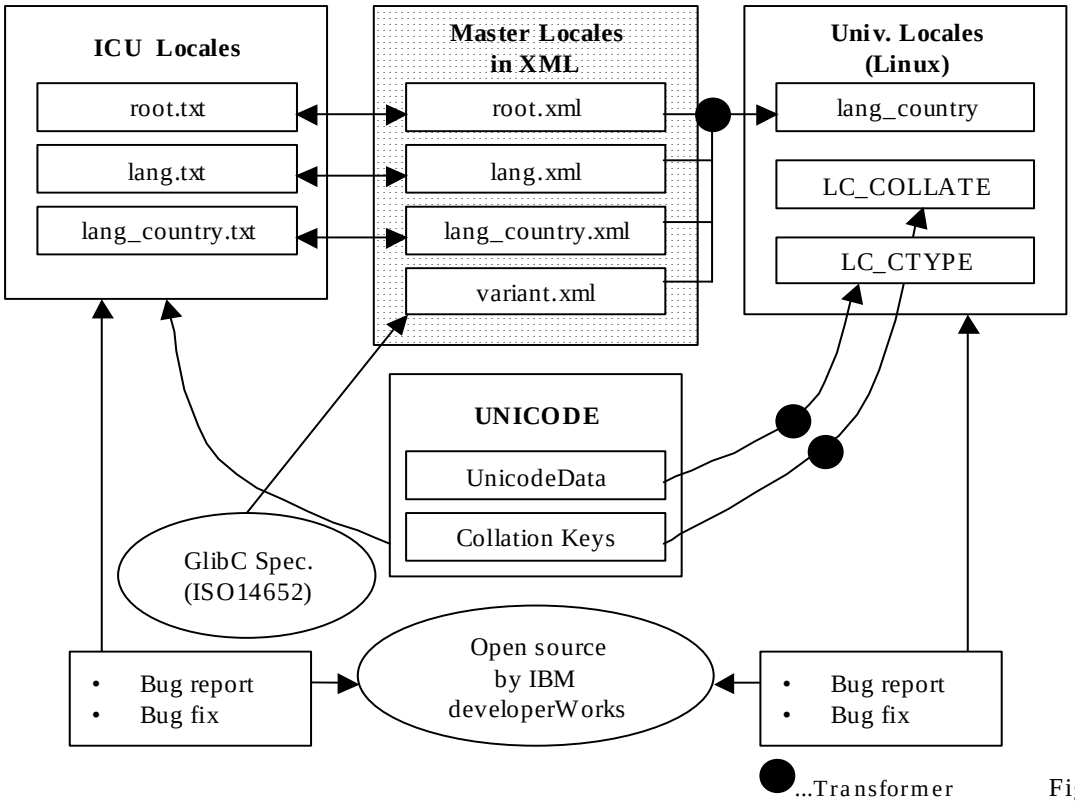


Fig. 1

2.2 Universal Locales for Linux

The Universal Locales for Linux follows the LI18NUX 2000 Globalization specification V1.0 that defines the interfaces and functions that must be supported by operating systems to run internationalized application software. The basis of this specification is from POSIX, XPG, Java and ICU. Annex B (Normative) to this document introduces the supported locales and code sets. The Universal Locale for Linux supports all locales on this list but the supported encoding is UTF-8 only. The Universal Locales makes use of the UCA, and the character classification in the Unicode Character database that is specified by the Unicode Standard.

2.3 ICU and Java

The ICU is consists of Unicode-based internationalization components for various platforms. It provides globalization functionalities such as locale-related functions, resource isolation, code conversion, collation, transliteration, word, line and sentence breaks. The Universal Locales for Linux was developed to be equivalent to the ICU locale data. The Universal Locales for Linux are equivalent to Java's locale-related functions because Java uses the ICU architecture. Fig. 2 indicates the relation among ICU, Java and Linux.

Fig. 2

2.4 GlibC V2.2 internationalization specifications

The GlibC V2.2 internationalization specification refers not only to POSIX/XPG but also to the ISO/IEC 14652 [11]-Specification method for cultural conventions. The ISO/IEC 14652 is an expansion of the specification of POSIX/XPG. It is revised frequently because it is still a draft, but the GlibC V2.2 has already implemented the following locale categories.

- Identification of the locale … LC_IDENTIFICATION
- Name format and data … LC_NAME
- Address format and data … LC_ADDRESS
- Telephone format and data … LC_TELEPHONE

We provide this locale data as supplemental data. They are filled in using the IBM National Language Design Guide Vol. 2 [12]

3. The design of Locale data in XML

3.1 Locale data in XML

In our locale development process, all Linux locales are transformed from the master locale data file. The master locale data is represented in XML because XML is an open standard and has the flexibility to exchange data among the various structured data repositories. The flexibility of data exchange is a very important property for our project because;

1. We created only Linux locales at this time, but the master Locales should have the capability to be transformed to any kind of locales such as Linux, ICU, Java, POSIX, and so on.
2. The master locales need to be integrated with the ICU locale data, UNICODE collation and character class, and the IBM National Language Design Guide.

We use XML for the master locale definitions in order to meet the above two requirements. The design of the locale data in XML is based on the ICU locale architecture. The majority of the syntax used to describe locale data is the same as the syntax of ICU locale data. Although the GlibC specific locale categories are not defined on the ICU, the master locales include them as supplemental locale data in XML. Their syntax follows ICU. This locale data in XML is provided by the ICU development group.

Fig. 3 is an example of locale data in XML. This example shows both the number and currency format. The locale data format is represented in a pattern. The first decimal pattern " #,##0.###;-#,##0.### " indicates that the negative sign is " - " and the grouping is by groups of three digits.

```
<numberFormat class="decimal" default="true">
  <patterns>
    <decimal>#,##0.###;-#,##0.###</decimal>
    <percent>#,##0%</percent>
    <scientific>¤#,##0.00;( ¤#,##0.00)</scientific>
  </patterns>
  <symbols>
    <decimal>.</decimal>
    <group>,</group>
    <list>;</list>
    <percentSign>%</percentSign>
    <nativeZeroDigit>0</nativeZeroDigit>
    <patternDigit>#</patternDigit>
    <plusSign>+</plusSign>
    <minusSign>-</minusSign>
    <exponential>E</exponential>
    <perMille></perMille>
    <infinity></infinity>
    <nan></nan>
  </symbols>
</numberFormat>
<currencies>
  <currency id="USD" default="true">
    <symbol>$</symbol>
    <name>USD</name>
    <decimal>.</decimal>
    <pattern>¤#,##0.00;( ¤#,##0.00)</pattern>
  </currency>
</currencies>
```

Fig. 3

3.2. Locale categories in XML

The locale category in the master locale data in XML should be a superset of the Linux locales, ICU, POSIX and so on because the master locale should hava the capability to generate various locales. Table 1 shows the locale comparisons between categories for Linux, ICU, and POSIX. The common categories in this table are three categories: calendar, monetary, and numeric. If the superset of categories on this table is included in the master locales, character classifications and full collation tables should be in the locale data in XML. However, neither is included because they are provided within Unicode.

Apart from the full collation table, the locale sensitive collation table is specified as the tailored collation category in the master locale data in XML. The tailoring syntax follows the ICU collation.

In summary, the locale categories in XML are below.

- Calendar
- Numeric
- Monetary
- YesNo Message
- Address
- Telephone
- Paper size
- Time zone
- Tailored collation

The transliteration category is currently being discussed.

TABLE 1
Locale category comparison

Platform		Linux (GlibC)	POSIX	ICU (V1.7)
Hierarchy				X *3
Category	CALENDAR	X	X	X
	MONETARY	X	X	X
	NUMERIC	X	X	X
	CHARACTER CLASSIFICATION	X	X	*4
	COLLATION	X	X	X *5
	YESNO MESSAGE	X	X	
	ADDRESS	X		
	TELEPHONE	X		
	PAPER SIZE	X *1		
	TRANSLITERATION	X *2		*6
	TIMEZONE			X

*1 PAPER SIZE is omitted from the latest ISO 14652(2000,11/15)
*2 Transliteration is added in the latest ISO 14652(2000,11/15)
*3 Locale in ICU inherits the parent locale data.
*4 Character class uses Unicodedata
*5 ICU V1.8 Collation implements UCA
*6 Transliteration is provided by another file, not locale (ICU V1.7)

4. The transformer from XML locale data to Linux locales

4.1 Transformer

The Transformer is a tool to generate Linux locales from the master locales in XML. The transformer parses the locale data in XML and generates Linux locale source file. It reads three kinds of file-locale data in XML, Unicode data, and collation key files. Both Unicode data and collation keys are provided within Unicode. The transformer has a capability option switch to set the output locale source files for Linux or POSIX.

The XML locale data files are hierarchical and this organization is followed by the ICU architecture. The ICU architecture specified a hierarchical locale tree framework. Fig. 4 illustrates the correspondence between the ICU locale tree and the XML locale tree. The root locale is located at the top of the locale tree. The language locale is a child of the root local, and the pair of language and country locale is the child of language locale. The transformer reads multiple XML locale files, and generates one united Linux or POSIX locale file. The transformer overwrite the locale data successively using the multiple XML files. Namely, a locale data is overwritten by the next XML file.

The output locale data is over by the last input locale data file from the XML.

The transformer generates a character classification for Linux that is categorized according to LC_CTYPE as well. The source file is UnicodeData.txt in the Unicode database. The output file is separated as a root character class file. Each Linux Locale source file copies it using the “copy” statement. Some of character classes are the locale sensitive even if they use Unicode-for example, the Turkish locale. These character classes require data that is generated from the modified source character class file. The modification is described on the SpecialCasing.txt file in the Unicode data.

The collation table for Linux that is used for LC_COLLATE is also created by the transformer, too. The transformer reads three kinds of collation file, basekeys.txt, compkeys.txt, and ctrkeys.txt as specified in the UCA. The generated Linux collation table is used as the root collation table, and each Linux Locale copies it using the “copy” statement.

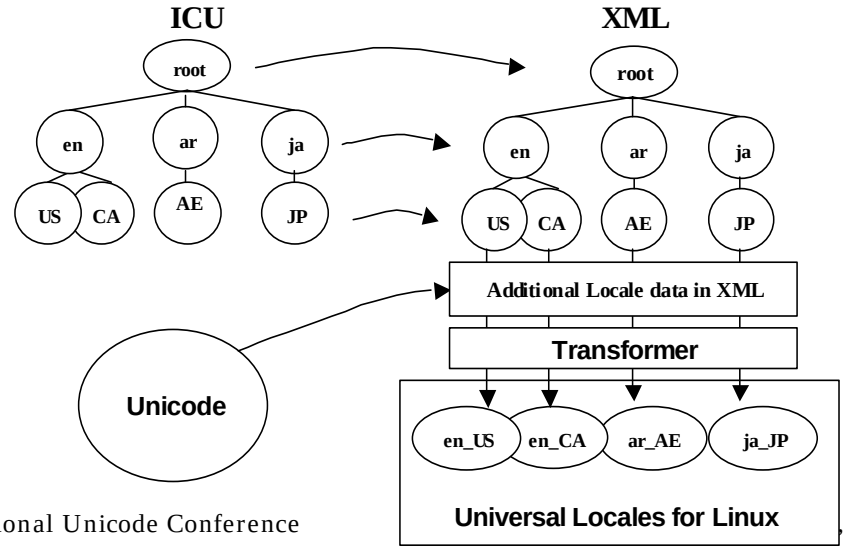


Fig. 4

4.2 Transformation of Locale data in XML

Table 2 shows the correspondence between each locale category in XML and each corresponding LC_*category* in the Linux locale system.

TABLE 2
The correspondence between locale categories in XML and Linux

XML		Linux		Description
Category	**Element**	**Category**	**Keyword**	
calendars	time, full	LC_TIME	t_fmt t_fmt_ampm	Time format
	date, full		d_fmt	Date format
	DateTime		d_t_fmt	Date and Time format
	monthNames		mon	Month
	MontheAbbr		abmon	Abbreviated Day
	DayNames		day	Day
	DayAbbr		abday	Abbreviated Day
	am pm		am_pm t_fmt_ampm	AmPm, Time format(AmPm)
numberFormat	decimal in the symbols	LC_NUMERIC	decimal_point	Decimal point
	Group in the symbols		thousands_sep	Thousand separator
		LC_MONETARY	mon_thousands_sep	Thousand separator for Monetary
	Decimal in the patterns		Grouping	Grouping for digit
currencies	Currency id	LC_MONETARY	int_curr_symbol	International currency symbol
	symbol		currency_symbol	Currency symbol
	decimal		mon_decimal_point	Decimal pointer for monetary
	pattern		positive_sign negative_sign int_frac_digits frac_digits p_cs_precedes p_sep_by_space n_cs_precedes n_sep_by_space p_sign_posn n_sign_posn	Monetary format
paperSize	Height	LC_PAPER	height	Paper size
	Width		width	
messages	yes	LC_MESSAGE	yesstr, yesexpr	Yes string and Regular expression
	no		nostr, noexpr	No string and Regular expression
	yesShort		yesexpr	Yes regular expression
	noShort		noexpr	No regular expression
AddressFormat	PostalPattern	LC_ADDRESS	postal_fmt	Address format
NameFormat	Namepattern	LC_NAME	name_fmt	Salutation

	GeneralSalutation		name_gen	
	ShortSalutationMr		name_mr	
	ShortSalutationMiss		name_miss	
	ShortSalutationMrs		name_mrs	
TelephoneFormat	InternationalPattern	LC_TELEPHONE		
	DomesticPattern			

As this paper already mentioned in Section 4.1, the locale data in XML is described using a pattern. Therefore, the transformer must change the locale data from a pattern to POSIX style syntax. Fig. 5 shows how the transformer changes the decimal pattern into the POSIX style.

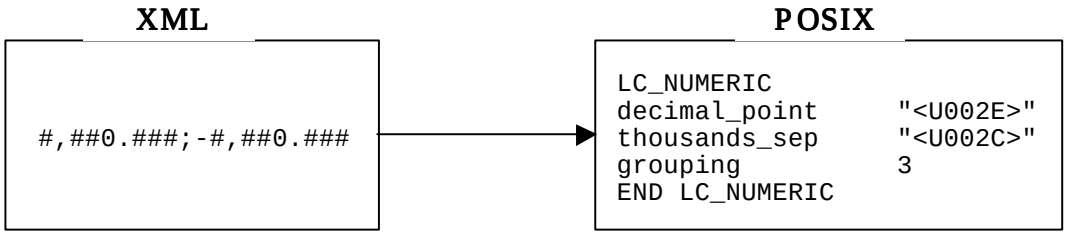


Fig. 5

4.3 Transformation of UNICODE collation keys to a POSIX style collation table.

The collation data in the master locale is maintained as files of collation keys in the UCA standard. The following is an extract from the basekeys file.

```
0F83 ; [*0000.0000.0000.0F83] # TIBETAN SIGN SNA LDAN
0F86 ; [*0000.0000.0000.0F86] # TIBETAN SIGN LCI RTAGS
0F87 ; [*0000.0000.0000.0F87] # TIBETAN SIGN YANG RTAGS
FEFF ; [*0000.0000.0000.FEFF] # ZERO WIDTH NO-BREAK SPACE
0009 ; [*0201.0020.0002.0009] # HORIZONTAL TABULATION (in 6429)
000A ; [*0202.0020.0002.000A] # LINE FEED (in 6429)
000B ; [*0203.0020.0002.000B] # VERTICAL TABULATION (in 6429)
000C ; [*0204.0020.0002.000C] # FORM FEED (in 6429)
000D ; [*0205.0020.0002.000D] # CARRIAGE RETURN (in 6429)
2028 ; [*0206.0020.0002.2028] # LINE SEPARATOR
2029 ; [*0207.0020.0002.2029] # PARAGRAPH SEPARATOR
200B ; [*0208.0020.0002.200B] # ZERO WIDTH SPACE
0020 ; [*0209.0020.0002.0020] # SPACE
```

Fig. 6

The transformer reads the files of collation keys, basekeys.txt, compkeys.txt, and ctrkeys.txt which are provided according to the UCA standard, and generates a POSIX style collation table. The transformer generates the following POSIX type collation definition shown in the Fig. 7 from the table in Fig. 6

```
<U0F83>   IGNORE;IGNORE;IGNORE;<U0F83>   # TIBETAN SIGN SNA LDAN
<U0F86>   IGNORE;IGNORE;IGNORE;<U0F86>   # TIBETAN SIGN LCI RTAGS
<U0F87>   IGNORE;IGNORE;IGNORE;<U0F87>   # TIBETAN SIGN YANG RTAGS
<UFEFF>   IGNORE;IGNORE;IGNORE;<UFEFF>   # ZERO WIDTH NO-BREAK SPACE
<U0009>   <S0201>;<D0020>;<C0002>;<U0009>      # HORIZONTAL TABULATION (in 6429)
<U000A>   <S0202>;<D0020>;<C0002>;<U000A>      # LINE FEED (in 6429)
<U000B>   <S0203>;<D0020>;<C0002>;<U000B>      # VERTICAL TABULATION (in 6429)
<U000C>   <S0204>;<D0020>;<C0002>;<U000C>      # FORM FEED (in 6429)
<U000D>   <S0205>;<D0020>;<C0002>;<U000D>      # CARRIAGE RETURN (in 6429)
<U2028>   <S0206>;<D0020>;<C0002>;<U2028>      # LINE SEPARATOR
<U2029>   <S0207>;<D0020>;<C0002>;<U2029>      # PARAGRAPH SEPARATOR
<U200B>   <S0208>;<D0020>;<C0002>;<U200B>      # ZERO WIDTH SPACE
<U0020>   <S0209>;<D0020>;<C0002>;<U0020>      # SPACE
```

Fig. 7

4.4 Transformation from UNICODE DATA to POSIX style character classification

The character class for the master locale is maintained in the UnicodeData.txt file in the Unicode data [10]. Fig. 8 is an extract from the actual UnicodeData.txt file that is included in the Unicode data.

```
0020;SPACE;Zs;0;WS;;;;;N;;;;;
0021;EXCLAMATION MARK;Po;0;ON;;;;;N;;;;;
0022;QUOTATION MARK;Po;0;ON;;;;;N;;;;;
0023;NUMBER SIGN;Po;0;ET;;;;;N;;;;;
0024;DOLLAR SIGN;Sc;0;ET;;;;;N;;;;;
0025;PERCENT SIGN;Po;0;ET;;;;;N;;;;;
0026;AMPERSAND;Po;0;ON;;;;;N;;;;;
…
003D;EQUALS SIGN;Sm;0;ON;;;;;N;;;;;
003E;GREATER-THAN SIGN;Sm;0;ON;;;;;Y;;;;;
003F;QUESTION MARK;Po;0;ON;;;;;N;;;;;
0040;COMMERCIAL AT;Po;0;ON;;;;;N;;;;;
0041;LATIN CAPITAL LETTER A;Lu;0;L;;;;;N;;;;0061;
0042;LATIN CAPITAL LETTER B;Lu;0;L;;;;;N;;;;0062;
0043;LATIN CAPITAL LETTER C;Lu;0;L;;;;;N;;;;0063;
```

Fig. 8

Table 3 explains the character attribute symbols and their descriptions. They are used in the UnicodeData.txt file.

TABLE 3
The character attribute on the Unicode data general category

Symbol	Description		
Lu	Letter, Uppercase	Lm	Letter, Modifier
Ll	Letter, Lowercase	Lo	Letter, Other
Lt	Letter, Titlecase	Pc	Punctuation, Connector
Mn	Mark, Non-Spacing	Pd	Punctuation, Dash
Mc	Mark,Spacing Combining	Ps	Punctuation, Open
Me	Mark, Enclosing	Pe	Punctuation, Close
Nd	Number, Decimal Digit	Pi	Punctuation, Initial quote
Nl	Number, Letter	Pf	Punctuation, Final quote
No	Number, Other	Po	Punctuation, Other
Zs	Separator, Space	Sm	Symbol, Math
Zl	Separator, Line	Sc	Symbol, Currency
Zp	Separator, Paragraph	Sk	Symbol, Modifier
Cc	Other, Control	So	Symbol, Other
Cf	Other, Format		
Cs	Other, Surrogate		
Co	Other, Private Use		
Cn	Other, Not Assigned		

The character classification and case conversion for the GlibC and POSIX/XPG locales are specified by the LC_CTYPE category. It has 11 classes-upper, lower, alpha, digit, space, cntrl, punct, graph, print, xdigit, blank, toupper and tolower. The Unicode data also provides the character classification as Fig. 8 indicates. The classification on the LC_CTYPE is the equivalent to the four attribute on the Unicode data. They are general category class, uppercase mapping, lowecase mapping and titlecase mapping. The Unicode data specifies other classifications such as canonical combining classes, a bidirectional category and character decomposition mapping, but these categories are not necessary for the LC_CTYPE. Table 4 shows the mapping rule between the general category on the Unicode data and the character class on the POSIX. This mapping from Unicode properties to POSIX is not specified in the Unicode Standard. This is the proposed mapping rule that is discussed within the ICU development. The transformer automatically transforms the LC_CTYPE from the Unicode data that is shown on Fig. 8 using this mapping rule.

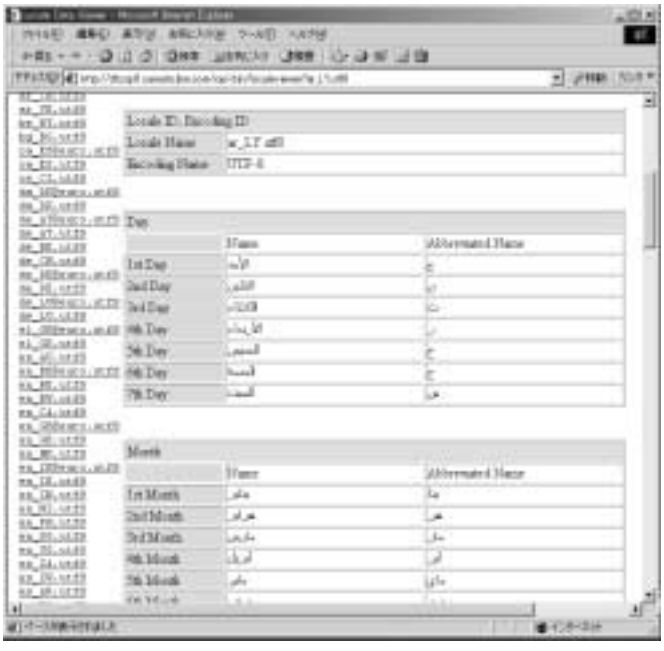
TBALE 4
Mapping rule between POSIX and UNICODE

POSIX	Unicode data
Upper	Lu, Lt
Lower	Ll
Alpha	L*, M*
Graph	L*, M*, N*, P*, S*
Print	Graph, Zs
Space	Z*
Blank	Zs, Zt
Contrl	Cc, Cf
Punc	P*, S*
Digit	Nd
Xdigit	0-9, A-F, a-F

Zt: TAB
Zb: CR, LF, FF, NL

5 Locale walker

The locale walker is a tool to browse the locale data, verify the collation, character classification, and case conversion. It is web-based CGI program written in C. Users are able to access the locale walker via Internet. Fig. 9 is a screen captured from the locale walker. It displays the locale data such as month name, day name, and so on.



The locale walker is good data verification tool for persons who are not familiar with locale, Linux, or internationalization. Using locale walker, any native speaker is able to verify the locale data, collation, and character classification easily, and easily join our open source project and contribute in ways open source such as bug reporting.

Fig. 9

6. Conclusion

In this paper, we introduce our project-Universal Locales for Linux-, its development process, and the development methodology using existing internationalization and Unicode resources. This paper also explains the locale data in XML, the Transformer and Locale Walker. By applying open source development methods to locale development and using existing resources, we were able to deliver over 140 Linux locales within 3 months, and also certify high quality based on the open source contributors' test reports. The Locale data in XML potentially has great expressive power for describing locale data. Locale data is not different among platforms or venders because it depends only on the language and the country. IBM is planning to use this file for the master locale database and would like to propose that the Unicode Consortium include it as part of the Unicode repository. The transformer is a Java program using XML for Java and has the capability to generate both Linux and POSIX locale representations.

Acknowledgments

We wish to thank Ulrich Drepper (glibc development) for his advice. He gave us many suggestions to improve this locale package, and pointed out many problems in the beta version. We also benefited from the efforts of the entire ICU development team. They provided many great i18n resources to reach the open source community. Raghuram Viswanadha (ICU development) provided us with the XML ICU locale data. Helena Shih (ICU development manager) defined the architecture of the XML locale data. Mark Davis (Unicode president) gave us a lot of advice and provided the XML-based UnicodeData. He also suggested a mapping rule between POSIX character classes and the Unicode Character Database. R. Hari (IBM India) provided us with the Telgu Indian Locale data. Debasish Banerjee (IBM WebSphere development) helped us with the Bengali India Locale.

Reference

[1] Unicode V3.0: http://www.unicode.org
[2] Unicode Collation Algorithm: http://www.unicode.org/unicode/reports/tr10/
[3] Unicode Online Data: http://www.unicode.org/unicode/onlinedat/online.html
[4] ISO/IEC 9945-2 ANSI/IEEE Std 1003.2 Information Technology –Portable Operating System Interface (POSIX) : Ref number ISO/IEC 9945-2:1993(E)
[5] XPG 4 System Interface Definitions, Issue 4, X/Open Document number C204.
[6] LI18NUX2000 Linux Globalization Specification:
http://www.li18nux.org/root/LI18NUX2000/index.html
[7] Linux Internationalization Initiative: http://www.li18nux.net/
[8] GNU C Library: http://www.gnu.org/software/libc/libc.html
[9] International Components for Unicode:
http://oss.software.ibm.com/developerworks/opensource/icu/
[10] Universal Locales for Linux: http://oss.software.ibm.com/developerworks/opensource/locale/
[11] ISO Cultural convention specification (draft ISO/IEC Technical Report 14652)
[12] *IBM National Language Design Guide* Vol 2. Pub. Number SE09-8802-03

Notices