FreeBSD, X-Windows, and I18N

Presented by (in alphabetical order):
Chia-Liang Kao <clkao@CirX.org>,
Clive Lin <clive@CirX.org>,
Michael Chin-Yuan Wu <keichii@iteration.net>

Topics Discussed:

Extract

This presentation discusses I18N and L10N in FreeBSD, X-Windows, and modern UNIX-style operating systems. It covers only the introduction level ideas. The paper also discusses proposals and hopes for future I18N development projects.

1. Introduction

This presentation discusses I18N, L10N in FreeBSD and X-Windows.

1.1. What is I18N and L10N?

I18N stands for internationalization, a common way to refer to the process of adapting modern operating systems in an international environment. (The word "internationalization" has 18 letters between the first "i" and the last "n," and it is unclear about who coined such a scheme of making acronyms.)

L10N stands for localization, with the similiary shortening scheme as I18N. L10N usually means taking I18N to the next level, making userland applications to appear entirely in certain languages.

1.2. What is XIM?

XIM stands for the X Input Method protocol, the X Consortium protocol defining the communication for "input methods" between XIM clients and servers. The writing languages of CJK are character-based, unlike those languages whose ‘‘words’’ are made up with ‘‘letters’’. Each character in CJK is unique. For instance, there are about 5000 characters that are frequently used. A typical font package for traditional Chinese would contain about 13000 characters. Obviously each character must be mapped to a sequence of key combination in order to be inputted. An methodology of the encoding mentioned is often called ‘‘input method’’. In most cases they are either by the pronunciation or shape, of the character.

1.3. What is Unicode?

Unicode is a character set that supposedly contains all of the necessary characters needed by the worlds' languages.

1.4. What are Locales?

The POSIX standard defines locales to be a geopolitical place or area, especially in the context of configuring an operating system or applications with its character sets, date and time formats, currency formats, etc.

From setlocale(3):

```
     LC_ALL       Set the entire locale generically.

     LC_COLLATE   Set a locale for string collation routines.  This controls
                  alphabetic ordering in strcoll() and strxfrm().

     LC_CTYPE     Set a locale for the ctype(3),  mbrune(3),  multibyte(3) and
                  rune(3) functions.  This controls recognition of upper and
                  lower case, alphabetic or non-alphabetic characters, and so
                  on.  The real work is done by the setrunelocale() function.

     LC_MESSAGES  Set a locale for message catalogs, see catopen(3) function.

     LC_MONETARY  Set a locale for formatting monetary values; this affects
                  the localeconv() function.

     LC_NUMERIC   Set a locale for formatting numbers.  This controls the for-
                  matting of decimal points in input and output of floating
                  point numbers in functions such as printf() and scanf(), as
                  well as values returned by localeconv().

     LC_TIME      Set a locale for formatting dates and times using the
                  strftime() function.
```

Common variables that need to be set by the user are LANG, LC_ALL, LC_CTYPE, LC_MESSAGES, and another related variable, MM_CHARSET. By the POSIX standard, if LANG or LC_ALL is set, all of the LC_* variables should automatically assumed to be the same as LANG or LC_ALL unless otherwise set by the user. Unfortunately, many programs do not follow this behavior and thus create problems for users and developers alike.

1.5 What are CJK?

CJK stands for Chinese, Japanese, and Korean in alphabetical order. Sometimes the V in Vietnamese is added to the acronym and becomes CJKV. The CJK languages use glyphs, contain tens of thousands of glyphs and are unmappable to European alphabets. Hence, CJK charsets use at least 8-bits in encoding instead of 7-bit encodings of European languages, creating many problems to applications written to use 7-bits. (e.g., telnet(1))

2. Kernel, Basesystem I18N

2.1. ISO/IEC/ANSI/POSIX Standards

The usual organizations that govern the computer engineering industry makes standards for I18N also. Should you wish for further information, please find the related documents from the governing organization.

2.2. Filesystems

This section discusses the progress of various works in the filesystems area of FreeBSD.

2.2.1. Unicode FFS

Michael C. Wu (one of the presenters) is currently working on changing the Berkeley Fast Filesystem to use the UNICODE charset by default. However, because many parts of the FreeBSD distribution were written with the assumption that the filesystem is simple ASCII, all of these parts will need to be changed before such a goal could be attained.

The implementation is still at its infant stages. Basically, upon completion, FFS should store all of its filenames in raw unicode. When the system requests a file, the kernel looks up the locale set by the user and returns the filenames in the correct charset after ICONV'ing from UNICODE to the specified charset.

2.2.2. MSDOSFS, SMBFS, and NTFS

Despite repercussions about using commercial filesystems, the Microsoft implementations of I18N filesystems are the best available currently. Boris Popov <bp@freebsd.org> is working on SMBFS system that will be able to present charset filenames. Althought here are already several different dirty patchsets to FreeBSD's MSDOSFS for various charsets, many developers feel that having a general solution would be best for the future development and maintenance of FreeBSD. The FreeBSD NTFS implementation is not able to read the newer UNICODE NTFS and we hope to improve that in the future.

2.2.3. ISO9660 CDROM Formats and DVD Formats

FreeBSD lacks I18N support in these filesystems, having only a partial implementation. Programmers should avoid assuming that the support exists.

2.3. ICONV

ICONV is a library of functions that converts various character sets to and from each other.

Ongoing work in ICONV by Konstantin Chuguev <Konstantin.Chuguev@dante.org.uk> is pivotal to I18N in any area in FreeBSD. The base system needs a general interface to converting character sets.

2.4 libxpg4, wchar*, and setlocale(3)

FreeBSD currently lacks a good libxpg4, and has a patchset not in the source tree that implements the ANSI C wchar* functions. Jeroen Ruigrok van der Werven <asmodai@freebsd.org> is working on an implementation of the xpg4 libraries.

3. Userland Applications

3.1. Default FreeBSD Distribution Binaries

Many parts of /bin, /sbin, /usr/bin, and /usr/sbin is not able to display non-ASCII charsets. These programs need to be slowly modified by the I18N developers to allow for such functionality.

Example:
'ps auxwww|grep mpg123' while playing an mp3 file with a Chinese filename.

```
keichii   601 12.3  0.6  6108  748  p4  R+    \
12:28^[%/2BIG5-0^BBIG5-0^BU   \
0:04.30 mpg123 ../mp3/\     \
q\M-$\M-b/\M-%\M-n\M-(\M-U/\M-%\M-n\M-(\M-U - \M-.\M-v\M-$H\M-1\M-!\M-:q.mp3
```

```
'export LC_CTYPE=zh_TW.Big5 ; ls  /home/keichii/mp3/cmp3001/'
??????.????.mp3                  ???s.?????b????.mp3
??????.????.mp3                  ???s.?????E.mp3
???v??.?Z?H?q.mp3                ???s.?u???^??.mp3
???v??.????.mp3                  ???s.?P??.mp3
???v??.?A?^??.mp3                ?????F.?u????.mp3
???v??.???g????.mp3              ?w.mp3
```

(It is possible to display CJK charsets in modified xterm-subsitutes if one does 'export LC_CTYPE=en_US.ISO_8859-1 ; ls foo', which is not POSIX compliant.)

3.1. The FreeBSD Ports System

3.1.1. Current Implementation

Applications patched for different languages are stored in their respective language's directory. Users must be able to differentiate between two of the same ports to use the Port system effectively.

3.2. Works in Progress

3.2.1. ports/chinese/zh-i18n

Clive Lin and Michael C. Wu are working on a Port that works much like ports/x11/gnome to depend on many Ports to be installed for a fully functional traditional Chinese FreeBSD system. The Port will also include many configuration files necessary for a Chinese FreeBSD system. We hope to propose this as a standard for all languages and eventually import an option in sysinstall.

3.2.2. I18N Options for Respective Ports.

Due to the current limitations of the Ports system, the build process has no way of determining which port to use. We propose that bsd.ports.mk should be modified to detect a make.conf option to automatically build the correct language port.

3.3. The Future of DNS

The DNS authorities of the world are discussing the next generation of DNS. They have proposed that each language has its own domain mapped in each's character set. We urge programmers of networking applications to leave room for future development on such standards.

4. X-Windows and I18N

4.1. Programming I18N-compliant X-Windows Applications

Each X toolkit has its own I18N implementation. We recommend using the latest gtk or qt versions. However, one can create an I18N application based only on the X libraries. Please refer to the toolkits' documentations for details.

4.1.1. A Simple Example of I18N, X libs, and XIM

```
fontset = XCreateFontSet(display,base_font_list,,,) ;
/* Setting locale and hook XIM
   X I18N programming needs to be able to do setlocale(3).
   Ensure that X and libc supports setlocale(3).  XSupportsLocale
   is one implementation of such.  XSetLocaleModifiers hooks XIM
   to the user's XIM server as specified by the
   environment varible XMODIFIERS. The only @category supported well
   in X11R6 is @im. */

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

main()
{
        setlocale(LC_CTYPE, "");
        if (XSupportsLocale() != True) {
           printf("\n");
           exit(0);
        }
        /* Hook XIM only if XSupportsLocale success.*/
        XSetLocaleModifiers("");
}

/* FontSet */
    Before displaying (drawing) the multibytes words, we have to
    tell X what font we want. Here is the XCreateFontSet(3X11).
    Man XCreateFontSet for details.

Display *display;
XFontSet fontset;
char *base_fontlist="-*-iso8859-1,-*-" ;
/* We use -*-, X lib will choose proper font available fit to
   current locale */

char **missing_charset, *def_string;
int missing_charset_count;

fontset = XCreateFontSet(display, base_fontlist,
                                  &missing_charset_list,
                                  &missing_charset_count,
                                  &def_string);

/* Drawing the font:
   XmbDrawImageString(3X11) and XwcDrawImageString(3X11) draws the fonts.
   If your string is simple char *, use XmbDrawImageString(). If your
   string is wchat_t *, use XwcDrawImageString(). No special skills
   here. Just use those 2 functions above instead of XDrawImageString()
   and XDrawImageString16().
*/
```

4.2. The Concept of Fontsets

A concept called 'font set' is introduced. A fontset contains fonts from different character sets. For example, from my .gtkrc:

```
style "gtk-default-zh-tw" {
  fontset = "-adobe-helvetica-medium-r-normal--12-*-*-*-*-*-iso8859-*,\
             -default-kai-medium-r-normal--16-*-*-*-*-*-big5-0
}
```

The application should show text in English with -helvetica font and Chinese with -kai font, with proper locale setup. While developers may need different initialization for locale different toolkits.

As previously mentioned, Asian characters are mostly of large number, thus using true type font is strongly suggested because having fonts for all sizes is not economic.

Developers may want to specify fonts for certain purposes. In most cases, fixed fonts, which annoys I18N users. A mechanism to define fontsets with names is important. So that developers could use the name ''fixed'' or ''variable'' or others, to avoid the distubance of I18N harmony.

4.3. XIM

Before the adoption of XIM, there were two kinds of mechanism for inputing CJK text: embedded input method and private protocol.

Embedded input method applications, for example, CXTerm, have their own mechanism for synthesizing characters built-in, and unusable by other applications.

Private protocols, for example, xcin 2.3 and earlier, uses other mechanism provided by the X protocol, for example, XAtom, to communicate to applications for character synthesis. Application understanding the private protocol could work fine, like xcin2.3 with crxvt. But most applications not specifically developed for the private protocol will not work without some hack. An example of such a hack is XA+CV, which is basically a library preloaded to override Xlib functions to take care about xcin2.3 protocol.

XIM unifies the communication for input method, which makes developers without too much knowledge about i18n easily write i18n ready applications. In the Xlib level, please refer to the book ''X window Programming Manual, developers' supplement for R6''. for widget toolkits, please see the following section.

4.3.1. XIM Internals

An application providing the input method service is called the XIM server. Applications needs input method service are called XIM clients. But in the view of XIM server, their are just different ICs(input context). There might be not only one text field in an XIM client, each of them is called a input context. Each input context has its own context, including, the characters inputed for unfinished synthesis, and buffer for phrase-based input methods, etc.

4.3.2. XIM Applications

XIM is implemented in most modern toolkits, including Motif, GTK, QT, and is integrated in widgets for inputing text. So developers don't really have to worry about this when using these toolkits normally.
Leaving Room for I18N

Programming I18N software is quite easy, contrary to the general misconception. Many X toolkits already provide the interface and the API to do so. Frequently, a piece of software only requires wrapping the displayed strings inside a fontset function. Internationalizing the software would simply be extracting the strings used in the program and keeping translation in a seperate file to be loaded later.

The fontset and the strings to be displayed are determined by the shell environmental variables in setlocale(3) set by the user or administrator. Frequently, we encounter software that respects only certain variables. (e.g. respecting LC_ALL but not LC_CTYPE) Such applications break POSIX compliance and create problems for I18N developers.

Programs that format text or require user input should not be coded with the assumption of using only a certain charset (such as ASCII). If the program is a X-Windows application, make sure that the XIM protocol is respected. The XIM protocol is well documented and implemented in all of the newer versions of popular X toolkits except for TK. If the program is a console application, simply ensure that the setlocale(3) variables are respected.

5. Conclusion

The internationalization of FreeBSD will be quite a painful and slow process. Due to the amount of legacy in the FreeBSD source tree and other contributed sources, we have to modify many expected behaviors in addition to adding new functionality to implement I18N. Also, we urge that programmers and developers design their software with the idea that non-English speaking people may use their software too.

The lack of standards to follow is quite a shortcoming and very much needed. Current POSIX and other standards were not very well designed. In the process of internationalizing FreeBSD, we wish to promote a generalized standard for open or closed source software. There has been talks of organizing an effort for the BSD's much like the KAME Project. In short, a few developers patching and making bad patches to software will not be a long term solution.

Despite the late start of the internationalization efforts in FreeBSD and the Open Source world, many completely functional internationalized systems can be demostrated by developers and users around the world.

Links of Documents and Related Materials:

The FreeBSD Project <http://www.freebsd.org>
XCIN XIM Implementation <http://xcin.linux.org.tw>
ANSI Standards <http://www.ansi.org>
The GTK Project <http://www.gtk.org>
POSIX Standards <http://www.posix.org>
Troll, Inc. and QT <http://www.troll.no>
UNICODE <http://www.unicode.org>
X Consortium <http://www.x.org>
The XFree86 Project <http://www.xfree.org>
CJKV Information Processing, by Ken Lunde, published by O'Reilly Books, ISBN #1565922247