

# Localizing GNU/Linux and XFree86

## A Thailand's Experience

Theppitak Karoonboonyanan  
thep@linux.thai.net

January 2004

### Abstract

This paper summarizes information gathered by the author during localizing GNU/Linux and XFree86 for Thai. It is by no means a complete reference nor a universal guide for all languages. The author just hopes it is useful for other localizers.

Internationalization is a practice commonly used for making software adaptable to local cultures without modification. Most of this paper discusses this framework and how to localize the system for new locales.

## Localizing GNU/Linux and XFree86

- Locale
- Overview of GNU/Linux Desktop I18N
- I18N in X11R6
- X Input Methods
- Font Systems

Note

## 1 Locale

### What is Locale?

- **Internationalization (I18N):** framework abstracting culture-dependent behaviors
- **Locale:** customization according to specific cultures
- Examples:
  - **POSIX Locale:** locale for C library
  - **GNU gettext:** locale for message translation

Note

## Locale Naming

OpenI18N:

```
<lang>-<territory>.<codeset>[@<modifiers>]
```

- *<lang>* = language code (ISO 639)
- *<territory>* = country code (ISO 3166)
- *<codeset>* = character encoding (if omitted, locale default is assumed)
- *<modifiers>* = locale modifiers

Examples:

- `fr_CA.ISO-8859-1`  
= French language spoken in Canada using ISO-8859-1 character set
- `th_TH.TIS-620`  
= Thai language in Thailand using TIS-620 encoding
- `th_TH`  
≡ `th_TH.TIS-620` in general

Note

## Character Sets

- GNU C library: Unicode-based
- Character sets described as Unicode subset

Example for UTF-8:

```
...
<U0041> /x41          LATIN CAPITAL LETTER A
<U0042> /x42          LATIN CAPITAL LETTER B
<U0043> /x43          LATIN CAPITAL LETTER C
...
<U0E01> /xe0/xb8/x81  THAI CHARACTER KO KAI
<U0E02> /xe0/xb8/x82  THAI CHARACTER KHO KHAI
<U0E03> /xe0/xb8/x83  THAI CHARACTER KHO KHUAT
...
```

Note

## Character Sets

Example for TIS-620:

```
...
<U0041> /x41    LATIN CAPITAL LETTER A
<U0042> /x42    LATIN CAPITAL LETTER B
<U0043> /x43    LATIN CAPITAL LETTER C
...
<U0E01> /xa1    THAI CHARACTER KO KAI
<U0E02> /xa2    THAI CHARACTER KHO KHAI
<U0E03> /xa3    THAI CHARACTER KHO KHUAT
...
```

Note

## POSIX Cultural Conventions

- Locale for the C language library

category	description
LC_CTYPE	character classification
LC_COLLATE	string collation
LC_TIME	date and time format
LC_NUMERIC	number format
LC_MONETARY	currency format
LC_MESSAGES	locale messages

Note

## POSIX Cultural Conventions

### Setting Locale

- C application:

```
/* <locale.h> */  
/* Set and/or return the current locale. */  
extern char *setlocale (int category,  
                        const char *locale);
```

For example:

```
#include <locale.h>  
...  
const char *prev_locale;  
prev_locale = setlocale(LC_ALL, "");
```

- empty-string locale means relying on environment setting:
  1. LC\_ALL defined → use it
  2. LC\_CTYPE, LC\_COLLATE, ..., LC\_MESSAGES defined → use them
  3. LANG defined → use it
  4. Otherwise → fall back to C (aka POSIX)

Note

## POSIX Cultural Conventions

### LC\_CTYPE

- C functions affected: <ctype.h>

isctrl()	isalnum()	isupper()
isgraph()	isalpha()	tolower()
isprint()	isdigit()	toupper()
isspace()	isxdigit()	
ispunct()	islower()	

- glibc: mostly relies on “i18n”

Note

## POSIX Cultural Conventions

### LC\_COLLATE

- C functions affected:
  - `strcoll()` = internationalized version of `strcmp()`
  - `strxfrm()` creates sorting key so that `strcmp()` yields the same order as applying `strcoll()` to source strings
- glibc: relies on ISO/IEC 14651 *Common Tailorable Template (CTT)* by default
  - localizers should check the “i18n” definition before tailoring it
  - Thai & Lao = exceptional cases
- locale definition:
  - multi-pass comparison → multi-level weight
  - ISO/IEC 14651 = 4 levels
  - describe weight as “IGNORE” to skip char

Note

## POSIX Cultural Conventions

### LC\_TIME

- C functions affected:
  - `strftime()`
- locale definitions:
  - day of week & month names translation
  - appropriate date & time formats
  - era

Note

## POSIX Cultural Conventions

### LC\_NUMERIC & LC\_MONETARY

- C functions affected:
  - `localeconv()` retrieves information for both categories
  - (non-standard) `strfmon()` format monetary amounts
- locale definitions:
  - LC\_NUMERIC: decimal point, thousand separator, number grouping
  - LC\_MONETARY: currency symbol (ISO 4217), monetary formats (debit/credit, numeric format)

Note

## POSIX Cultural Conventions

### LC\_MESSAGES

- C functions affected: *none*
- locale definitions: yes/no responses
- mostly used by *GNU gettext*

Note

## ISO/IEC 14652

ISO/IEC 14652 = extended POSIX, plus:

category	description
LC_PAPER	paper size
LC_NAME	personal name format
LC_ADDRESS	address codes and format
LC_TELEPHONE	telephone number
LC_MEASUREMENT	measurement units
LC_VERSIONS	locale version

- already supported by glibc
- info can be retrieved using `nl_langinfo()`  
(in fact, all locale data can be retrieved as such)

Note

## Building Locale

- Files needed
  - locale definition
  - charmap
- Symbolic names for char are usually Unicode-based
- To compile locale data:

```
localedef [-f <charmap>] [-i <input>] <name>
```

for example:

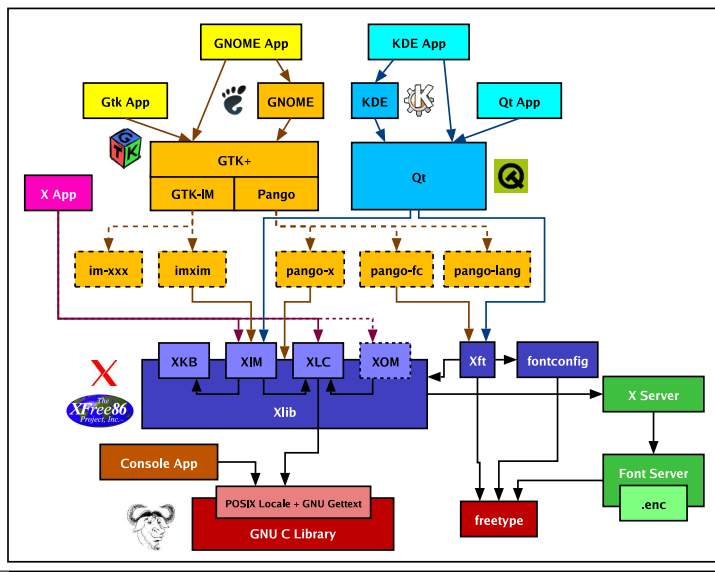
```
# localedef -f TIS-620 -i th_TH th_TH
```

- The `locale` command:
  - argument-less → list values as affected by env
  - “-a” → list installed locales
  - “-m” → list supported charmaps

Note

## 2 Overview of GNU/Linux Desktop I18N

### GNU/Linux Desktop



Note

## GNU/Linux Desktop

### GNU C library

- ISO/IEC 14652 locales (POSIX proper superset)

### XFree86 (X11R6.5)

- X locale (XLC)
- X Input Method (XIM)
- X Output Method (XOM)

Note

## GNU/Linux Desktop

### GTK+ 2

- Unicode-based Multilingual System
- GTK+ Input Method (gtk-immodules)
- Pango
  - rendering engine: X, Xft, OpenType backends
  - language engine: cursor movements, word break

### Qt 3

- Unicode-based Multilingual System
- XIM-based input method
- Unicode-based complex text support (QComplexText class)

Note



### 3 I18N in X11R6

#### I18N in X11R6

**Xlib:** I18N via X locales

- X locale components:
  - **XLC** – locale data
    - \* **XLC\_FONTSET** – font charset encoding, used by XOM
    - \* **XLC\_XLOCALE** – character encoding, conversion, etc.
  - **XIM** – input method
  - **XOM** – output method

Note

### 4 X Input Method

#### Keyboard Maps

**X11R6:** XKB Extension

- **XKB:** scan code  $\xrightarrow{\text{model}}$  keycode  $\xrightarrow{\text{layout, variant}}$  keysym
- **scan code:** raw scan code from kernel
- **model:** hardware-dependent  
(pc104, pc105, microsoft, microsoftplus, ...)
- **keycode:** symbolic values for *unscreened* key positions
- **layout:** language-dependent layout
- **variant:** sub-type of layout (mostly “*nodeadkeys*” for Latin)
- **keysym:** symbolic values for *screened* keyboard

Note

## Keyboard Maps

### Setting up XKB

- `setxkbmap` command
  - *rules*: XFree86 4.x default method
    - \* normally "xfree86"
    - \* rule files: `/etc/X11/xkb/rules`
    - \* rules: determine *layout*, *variant*, *option* to produce the symbols map
  - *model*: hardware model
  - *layout*: up to 64 groups mixture
  - *variant*: pick up variation
  - *option*: additional configurations
    - \* group switching key
    - \* level-3 shift
    - \* Ctrl and CapsLock rearrangement
    - \* LED indicator
    - \* CapsLock behaviors
    - \* etc.

Note

## Keyboard Maps

### Setting up XKB

- `setxkbmap` command (cont.)

Example:

```
$ setxkbmap us,th \  
-option grp:alt_shift_toggle,grp_led:scroll
```

Full options:

```
$ setxkbmap -rules xfree86 \  
-model pc104 -layout "us,th" \  
-option "grp:alt_shift_toggle,grp_led:scroll"
```

Note

## Keyboard Maps

### Setting up XKB

- /etc/X11/XF86Config:

```
Section "InputDevice"
  Identifier "Generic Keyboard"
  Driver      "keyboard"
  Option      "CoreKeyboard"
  Option      "XkbRules"      "xfree86"
  Option      "XkbModel"      "pc104"
  Option      "XkbLayout"     "us,th"
  Option      "XkbOptions"    "grp:alt_shift..."
EndSection
```

Note

## Providing a Keyboard Map

- New keymap → new symbols file  
(/etc/X11/xkb/symbols)
- Default symbols used by XFree86 4.3.0 default rules:  
/etc/X11/xkb/symbols/pc/\*  
(one group per file, no pre-combined)
- Symbols file:
  - xkb\_symbols data
  - entries:
    - \* group name
    - \* keysyms for each keycode (unshifted + shifted)
  - keysyms:
    - \* predefined (<X11/keysymdef.h>)
    - \* Unicode keysyms (0x100xxxx)

Note

## Providing a Keyboard Map

- Example: `/etc/X11/xkb/symbols/pc/th.tis`

```
partial default alphanumeric_keys
xkb_symbols "basic" {
    name[Group1]= "Thai (TIS-820.2538)";
    ...
    key <TLDE> {[0x1000e4f, 0x1000e5b]};
    key <AE01> {[Thai_baht, Thai_lakkhangyao]};
    key <AE02> {[slash, Thai_leknung]};
    ...
};
```

- Regenerate the symbols list:

```
# cd /etc/X11/xkb/symbols
# xkbcomp -lhlpr '*' -o ../symbols.dir
```

Note

## Providing a Keyboard Map

- Extra:
  - add entry to `/etc/X11/xkb/rules/xfree86.lst`
- XFree86 source:
  - `xc/programs/xkbcomp`

Note

## X Input Method

### XIM

- Locale-based framework for text input of any language
- Structure:
  - X client: Input Context (XIC) for text entry
  - XIM: service (server or library)
  - X client passes key events to XIM
  - XIM translate keycode and determine action upon keysym, based on current state of XIC
  - XIM commits character back to X client event queue when conversion is ready

Note

## X Input Method

### User's point of view

- In general, XIM-aware app calls `XSetLocaleModifiers("")`, which allows user to choose XIM via `XMODIFIERS` env.
- For example:

```
$ export LANG=th_TH.TIS-620
$ export XMODIFIERS="@im=Strict"
```

which specifies `Strict` input method for Thai locale.

Note

## 5 Font Systems

### Traditional X Font System

#### X Fonts

- BDF, PCF, Type 1, TrueType
- Owned and Managed by X server or X Font Server (XFS)
- Naming: X Logical Font Description (XLFD)

```
-adobe-times-medium-r-normal--*-100-75-75-p-54-iso8859-1
```

Note

## Traditional X Font System

### X Fonts

- Installing:
  - Prepare `fonts.dir` with `mkfontscale` & `mkfontdir`
  - Add the font path to either:

- \* X server (immediate):

```
$ xset fp+ /your/font/path
$ xset fp rehash
```

- \* X server (next startup):  
`/etc/X11/XF86Config`:

```
Section "Files"
...
FontPath "/your/font/path"
...
EndSection
```

- \* XFS: `/etc/X11/fs/config`:

```
catalog=...,/your/font/path
```

then:

```
# /etc/init.d/xfs restart
```

Note

## Xft and Fontconfig

**XFree86 4** introduces:

- **X Render Extension:** alpha, anti-aliasing, sub-pixel (server side)
- **Xft Library:** rasterize w/ FreeType & X Render (client side)
- **Fontconfig Library:** font accessing (client side)

→ Client-side fonts + Anti-aliased

**Xft/fontconfig**

- Used by GNOME 2 & KDE 3  
→ becomes default usage
- Traditional X fonts still required by GTK+ 1.2 apps, though

Note

# Xft and Fontconfig

## Configuration

- `/etc/fonts/fonts.conf`
  - XML format
  - `<dir>` lists font (root) directories
  - others: instruction for font matching & customization
- Common default config:
  - System fonts at `/usr/share/fonts/`
  - User fonts at `~/.fonts/`
- Installing fonts:
  - Copy fonts into one of the defined font paths
  - Regenerate font cache

```
# fc-cache -f
```

- Listing fonts:

```
# fc-list
```

Note