i18n on Linux by example: Japanese

Christian Horn

July 14, 2014

Table of contents

my background

- Christian Horn
- * in Mühlhausen, Thüringen/Germany
- Russian in school — I learn: I am bad at languages.
- later English — I confirm: I am bad at languages.
- Munich, work.
  - #linuxadmin #engineer #architect #typography
- 2008 for 3 months in Japan, since then: learning Japanese.
  - #language #culture #people

about this talk

- topic outline: i18n on linux, emphasis on Japanese
- What are the concepts, what typical issues are hit?
- This is me sharing what I find fascinating
- Let me know if it's not that interesting for others, then we directly head for beer :)

motivation of dealing with i18n/linux/Japanese

- To learn a language you want to use ways interesting to you: reading Japanese literature, watching TV series, ...
- You naturally integrate the language into your daily life: when reading news, writing mails..
- It's fun to look at obscure things.
  - Obscure for us in the English speaking IT world - but using Cyrillic or Kanji is not obscure in Asia.

What is i18n?

Is this taxi occupied or free?

What is i18n?

Is this taxi occupied or free?

It's free. Although a European mostly maps 'red' to 'not free'.

Why do we need i18n?

- Based on culture we make assumptions
- Here most Europeans conclude 'colour red, car is occupied'
- Adapting software to our reading habits and common understanding makes it easier for us to interact, less prone to errors.

Why do we need i18n?

Depending on culture, things are interpreted differently.

- × (ばつ, 'batzu')
  - means in Japan and other parts of the world 'incorrect'
- ◯ (まる, maru)
  - means in Japan 'correct'
  - is in Europe used to mark errors.
- ✓('check mark')
  - means 'correct' in most parts of the world

What is i18n?

- Can you imagine your grandparents using a computer not speaking their native language?
- What is the meaning of 2/6/14 ?
  Probably a date.. but whats month/day? All over the world it's read differently.
- We need i18n to interact with the system in the most efficient way. We understand icons with signs from our culture fastest, read local books fastest, we are most effective in our native language.

What are i18n and l10n?

- **internationalization/i18n:** building a system/software which can easily adapt to a variety of languages, cultures and customs. i.e. date and time formats, local calendars, number formats and numeral systems, handling of personal names, forms of address etc.
- **localization/l10n:** The actual adaption of documents, programs etc. to local languages and cultures. Includes translations, custom icons (for easier understanding), time formats, paper type, currency, weekday start etc.

The basics of written Japanese

Typical writings contain all of these:

1. **hiragana:** 「にほんごでかきましょう。」
   children
   start learning here. Contains the 5 vowels, plus 45 consonant/vowel unions like "ma", "ku" and such. All Japanese sentences and names can already be expressed with hiragana!
2. **katakana:** 「ニホンゴデカキマショウ。」
   used mainly for imported words and names. Contains again the 5 vowels, and 45 unions which sound the same as hiragana.
3. **kanji:** 「日本語で書きましょう。」
   pictogram based, many thousand exist.

人雲山天
犬霧川地
上室峰星
末苔谷空
ひくやあ
ともまめ
いきかつ
ぬりはち
うむみほ
へろねし
すこたそ
ゑけにら

Ametsuchi-No-Uta

encountering encodings

- So then you receive files with more exotic contents, i.e. textfiles with Japanese Kanji:

```
$ file myfile.txt
myfile.txt: UTF-8 Unicode text
```

- multiple topics coming up:
  1. identifying filetype/encoding
  2. displaying the file
  3. modifying or processing the file

character encodings: fixed width, worldwide use

- encoding: recipe to translate the bits into characters
- ASCII: 7bit, specified 1963, has english alphabet, numbers + nonprintables = 128 chars
- EBCDIC: 8bit, 1963, from IBM, with 6 mutually incompatible versions meant to succeed ASCII
- code page 437: the famous DOS page from IBM, these pages being 8bit and ontop of ASCII. I.e. extending for Greek, Esperanto etc. - but only one at a time
- KOI8-R: 8bit, for Russian, stays readable when reading 7bit: "Р у с с кий Т е кс т" in KOI8-R becomes "rUSSKIJ tEKST"
- UTF-32: fixed width unicode implementation

character sets, japanese relevant

JIS (Japan Industrial Standards), organization responsible for coded character sets (CCS) and encodings used in Japan.

- JIS X 0201: Jap. Industry Standard, character set with 7 and 8 bit encodings, first encoding in 1969, ASCII + katakana. Allows phonetic expressing, but no hiragana or Kanji
- JIS X 0208: character set from 1978, 16bit encoding, ~6800 pictograms: most common Kanji
  - widely used on Japanese mobiles nowadays, not unicode
  - also the standard of Aozora Bunko
- Unicode: all of these (apart of JIS X 0213) are part of Unicode 3.0.1.

character encodings: variable width

- Shift-JIS: supports several JIS X character sets
  - famous on 2chan board since 1999, 4chan ancestor
  - good supported on Windows since 3.1j (CP932 flavour)
  - many nonstandard custom variants, i.e. with emoticons for mobile
- EUC-JP: includes several JIS X, updated many times
  - widely supported in Unix/Linux
  - yet InternetExplorer supports only a subset
- ISO-2022-JP: mails with:

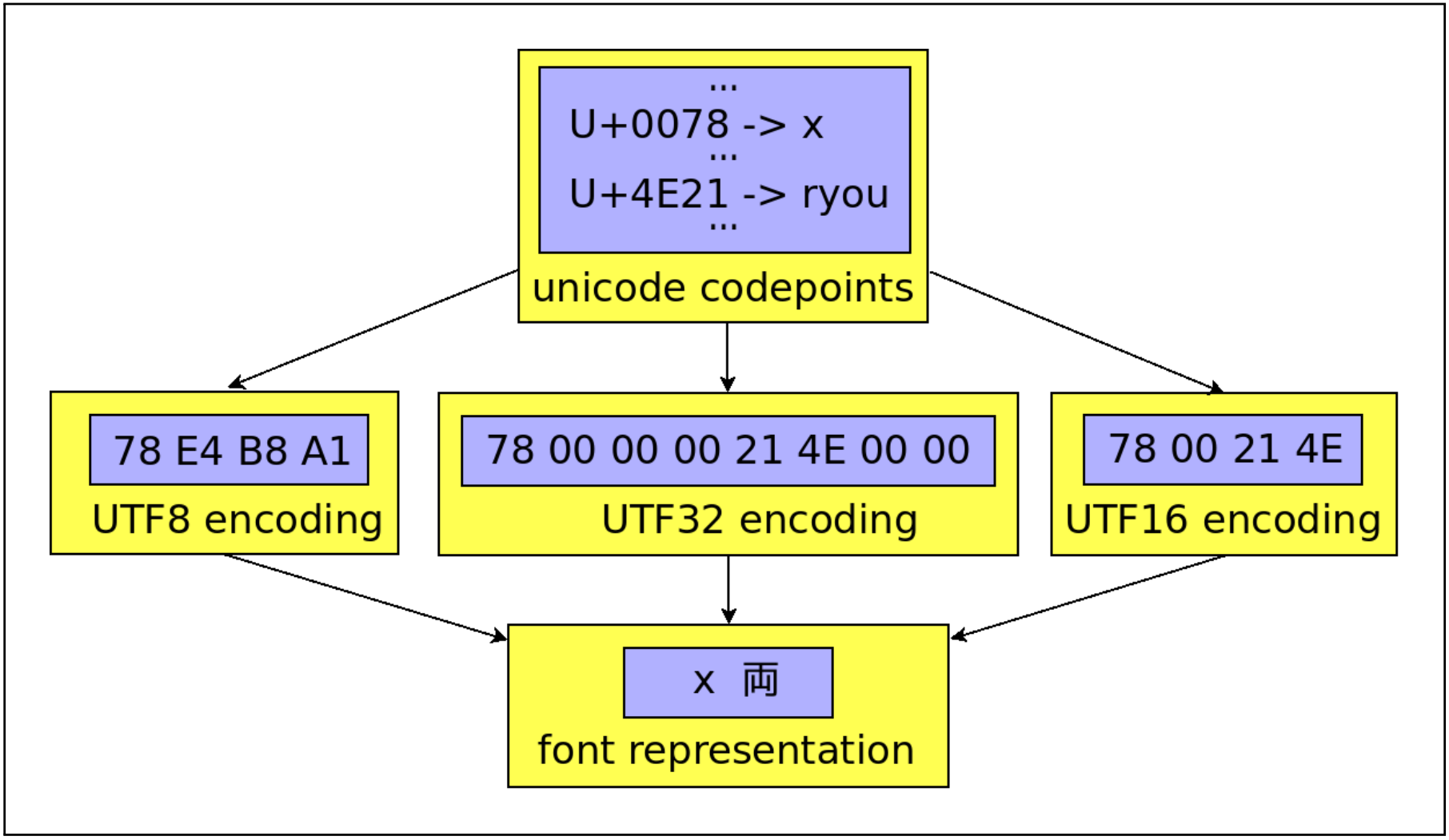
```
Content-Type: text/plain; charset="iso-2022-jp"
```

- UTF-8, UTF-16: implementations of Unicode standard, widely used in web (html4.0 recommendation) and mail (MIME)

unicode

- attempt to include all others
- defines only tables mapping codepoints (numbers) to character descriptions
- UTF-8, UTF-16, UTF-32 or UCS-2 are unicode encodings
- a file with ASCII chars is also valid UTF-8
- other encodings exist, i.e. UTF-7: guarantees that the 8th bit is always zero. For your not-so-clever transport system which might expect this
- HAN unification: effort to unify Chinese, Japanese, Korean kanji to aingle unified chars

unicode encodings



Fixed with encodings

- 'file' can be used
- resuling files have different length

```
$ echo "foobar123" >file_ascii
$ echo "foobar123" | iconv -f ascii -t utf32 >file_utf32b
$ ls -al file*
-rw-rw-r--. 1 chris chris 10 Jan 5 22:18 file_ascii
-rw-rw-r--. 1 chris chris 44 Jan 5 22:18 file_utf32
$ file *
file_ascii: ASCII text
file_utf32: Unicode text, UTF-32, little-endian
$ xxd file_ascii
0000000: 666f 6f62 6172 3132 330a foobar123.
$ xxd file_utf32
0000000: fffe 0000 6600 0000 6f00 0000 6f00 0000 ....f...o...o...
0000010: 6200 0000 6100 0000 7200 0000 3100 0000 b...a...r...1...
0000020: 3200 0000 3300 0000 0a00 0000 2...3.......
```

fixed vs. variable width encoding

```
$ echo -e '\x66\x6f\x6f\xe6\x97\xa5\xe6\x9c\xac' >f_utf8
$ iconv -f utf8 -t utf32 f_utf8 >f_utf32
$ file f*
f_utf32: Unicode text, UTF-32, little-endian
f_utf8: UTF-8 Unicode text
$ ls -al f*
-rw-rw-r--. 1 chris chris 28 Jan 5 22:50 f_utf32
-rw-rw-r--. 1 chris chris 10 Jan 5 22:50 f_utf8
$ xxd f_utf8
0000000: 666f 6fe6 97a5 e69c ac0a foo.......
$ xxd f_utf32
0000000: fffe 0000 6600 0000 6f00 0000 6f00 0000 ....f...o...o...
0000010: e565 0000 2c67 0000 0a00 0000 .e..,g......
```

$ LC_ALL=en_US.UTF-8 cat f_utf8
foo日本

variable width encoding examples

- Different characters of the same encoding can result in different length when encoded.

$ echo -n "日" >f_hi
$ echo -n "Я" >f_ya

```
$ file f_*
f_hi: UTF-8 Unicode text, with no line terminators
f_ya: UTF-8 Unicode text, with no line terminators
$ ls -al f*
-rw-rw-r--. 1 chris chris 3 Jan 5 23:31 f_hi
-rw-rw-r--. 1 chris chris 2 Jan 5 23:31 f_ya
$ xxd f_hi
0000000: e697 a5 ...
$ xxd f_ya
0000000: d18f ..
```

displaying UTF8 characters on Linux

1. Which utf8 locales are available?

   ```
   locale -a|grep utf8
   ```

2. Tell the system in which language to communicate:

   ```
   for i in LC_ALL LANG LANGUAGE; do
       export $i=ru_RU.utf8; done
   ```

3. Run a terminal emulator capable of utf8: xterm -en utf-8
4. Try to output utf8 character U+263a:

   ```
   perl -CSDL -le 'print "\x{263}"'
   ```

   Notice: also your font has to support the character! Program 'gucharmap' helps explore fonts.

How to convert between encodings?

Lets convert kanji into hiragana. We asume that 'kakasi' only speaks Shift-JIS. So we will:

- convert utf8 to shift-jis with iconv
- run kakasi
- convert the result to utf8, display it

```
echo "私は何も知らない。" | \
  iconv -f utf8 -t sjis | kakasi -JH | \
  iconv -f sjis -t utf8
わたしはなにもしらない。
```

kakasi update

kakasi 2.3.5 and later supports UTF-8 natively via "-i" and "-o" for input/output encodings

conversion of aosora documents to UTF-8

```
$ export LC_ALL=en_US.utf8
$ curl -O  http://www.aozora.gr.jp/cards/000081/files/
45630_txt_23610.zip
$ unzip 45630_txt_23610.zip
$ head -n 1 amenimo_makezu.txt
[ mojibake, it's shift-jis on utf-8 terminal ]
$ head -n 1 amenimo_makezu.txt | iconv -f sjis -t utf8
〔雨ニモマケズ〕
$ file amenimo_makezu.txt
amenimo_makezu.txt: Non-ISO extended-ASCII text, with
CRLF, NEL line terminators
$ lv amenimo_makezu.txt
```

Input of Russian or Japanese characters

So, do I need a keyboard with 5.000keys for Japanese?

IBM keyboard for Japanese characters, 1980, German museum

Input of Russian or Japanese characters

No 5000 keys keyboard required.
Cut'n'pasting text from websites does not scale either.

- the input method gets changed, switching i.e. between Japanese, Russian and English input (i.e. using strg+space)
- For Japanese Hiragana/Katakana can then directly be entered
- For Kanji we input the sound of the Kanji in Hiragana and have the input system complete to the desired Kanji
- romaji (i.e. "kan") → kana ("かん") → kanji ("館")
- many homophones, i.e. 61 Kanji readings for "kan"

Input of Russian or Japanese characters

- Input method frameworks:
  - ibus (Fedora, Ubuntu, FreeBSD, ..)
  - scim (BSDs)
  - fcitx (newcomer, very active development, many features)
- Input methods:
  - Anthy (old, not much movement),
  - mozc (more modern, handwriting tool, dictionary editor)
  - 'Simple Kana Kanji' (ibus-skk)
  - 'Kana Kanji engine' (ibus-kkc)

Locales: having programs act localized

- LOCALE concept introduced in ISO C (ISO/IEC 9899:1990), enhanced 1995
- also POSIX contains i18n standards
- locale categories:
  - LC_CTYPE: are multibyte chars used?
  - LC_COLLATE: sorting related
  - LC_MESSAGES: selects language of software output
  - LC_MONETARY: comma or period as separator, currency mark
  - LC_NUMERIC: numbers, character for dec. point etc.
  - LC_TIME: time, names of weekdays, date order etc.

Locales: example

So programs themself support multiple languages, lets use them:

```
$ for i in en_US aa_ER et_EE ru_RU uk_UA zh_CN ja_JP; do
>   LC_ALL=$i.utf8 date; done
Wed Jun 18 21:35:13 CEST 2014
Arbaqa, Qasa Dirri 18,  9:35:13 carra CEST 2014
K juuni 18 21:35:13 CEST 2014
Ср июн 18 21:35:13 CEST 2014
середа, 18 червня 2014 21:35:13 +0200
2014年 06月 18日 星期三 21:35:13 CEST
2014年  6月 18日 水曜日 21:35:13 CEST
```

i18n for email

- In the beginning, for 'From' etc. parts of the mail communication (SMTP) we use ASCII
- To use unicode i.e. in Subject, it gets encoded, i.e. Subject: Re: =?iso-8859-1?Q?n=E4chste?= Woche
- Content type instructs how to read the mail body ('data'): Content-Type: text/plain; charset="UTF-8"

i18n for webpages

- first stage: in the first part of the communication the webserver tells the content type:

```
Content-Type: text/html; charset=utf-8
Content-Type: text/plain; charset=utf-8
```

- second stage, in the transmitted document:

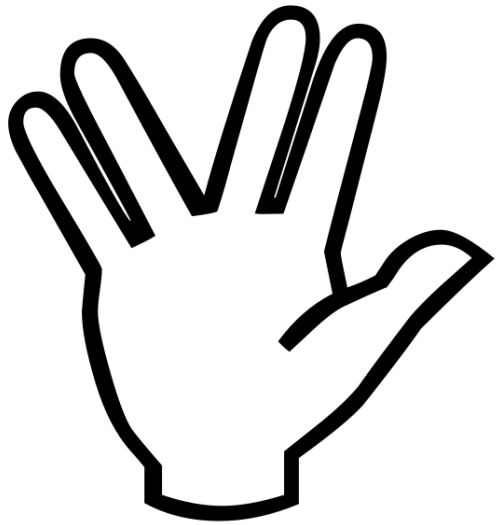
```
<!DOCTYPE html>
<html lang="en" dir="ltr" class="no-js">
<head>
  <meta charset="utf-8" />
  <title>start [fluxcoil.net]</title>
  ...
```

i18n: more software

- Japanese in libreoffice: possibly further fonts required, writing top-down and left-right possible?
- Japanese and LaTeX: CJK, works mostly. Yet, code boxes and UTF-8 are challenge
- mplayer w/ jap subtitles: command option for font used
- converting UTF8 texts from the net to vertical texts on the kindle paperwhite
  - PDF with kanji: OCR exists, FineReader Engine 11 CLI
  - tool 'luit': wrapper for terminal in EUC-JP, UTF-8 etc.
  - tool 'lv': smart less which detects encodings

links and sources

- Standing on shoulders of giants: great community around Linux and Japanese/i18n.
- tlug.jp Tokyo Linux user group
- Debians excellent "Introduction to i18n"
- Markus Kuhns UTF-8 and Unicode FAQ for Unix/Linux
- more links: get the LaTeXsources of this presentation



"Live long and prosper", U+1F596

http://fluxcoil.net
chorn@fluxcoil.net

i18n in UEFI, displaying unicode

- standard only enforces capability to display 'Basic Latin Unicode character set'
- the vendors UEFI on a system *might* already support to display the glyphs (character fonts) you want, i.e. Chinese.
- To be sure, ship the fonts for the unicode characters you need by yourself, together with the UEFI application. Details: 'Simplified Font Package' in the UEFI spec.
- these descriptions are for 8x19 pixel or 16x19 pixel glyphs, describe the fonts 'bitmap'.
- No prepackaged fonts are known so far. One could choose free fonts with appropriate license and convert the required glyphs for UEFI use

i18n in UEFI, input

- **UEFI internal code:**
  - internally exclusively unicode strings are used
  - UEFI editor handles utf-8 encoded files (displaying is different topic thou..)
- **input unicode:**
  - standard enforces only 'Basic Latin Unicode character set' to be supported for input in EFI console. It's a superset of ASCII, 16-bit chars.
  - any other unicode character set can optionally be supported.
  - So far no alternate implementations exist.

backup: possibly also interesting to mention

- And then you turn on "search in japanese, english and german". And you need help to stay effective in reading.
- firefox: rikaichan /rikaisama
- google translate. Also recognizing voice and handwritings, can read japanese loud, offers own input method for Japanese!

i18n on mobile/appliances

- Japanese input on mobile: easy, smart input methods exist also here!
- android.
  - "paint" kanji with the finger and get it recognized. offline/online.
  - say something and get it recognized! (rocket science!)
  - OCR of camera pictures of Kanji!
- Your tv might be able to play matroska, maybe even subtitles, but has probably not japanese fonts.