Supporting Japanese for Debian system

鵜飼 文敏 Fumitoshi UKAI
<ukai@debina.org>

後藤 正徳 Masanori GOTO
<gotom@debian.org>

Workshop menu

- Introduction to Japanese environment
  - Coding system (gotom)
  - Input Method (ukai)
- Undergoing projects
  - groff issue (ukai)
  - UTF-8 transition (gotom)
  - debconf I18N (ukai)

Introduction to Japanese environment

It’s not easy to use Japanese on Debian

- Many softwares can not handle multibyte characters.
- We have 4 encodings.
- We need Input Method software to input Japanese.
- Almost truetype fonts are non-free.

- Debian Japanese guys have been tried to improve many issues.

Japanese Coding System

- There are 4 different coding system
  - EUC-JP (Mostly we use on linux)
  - SJIS (Windoze/Mac - many people use it)
  - ISO-2022-JP (Mail)
  - UTF-8 (UCS-2)

How many characters in CJK characters

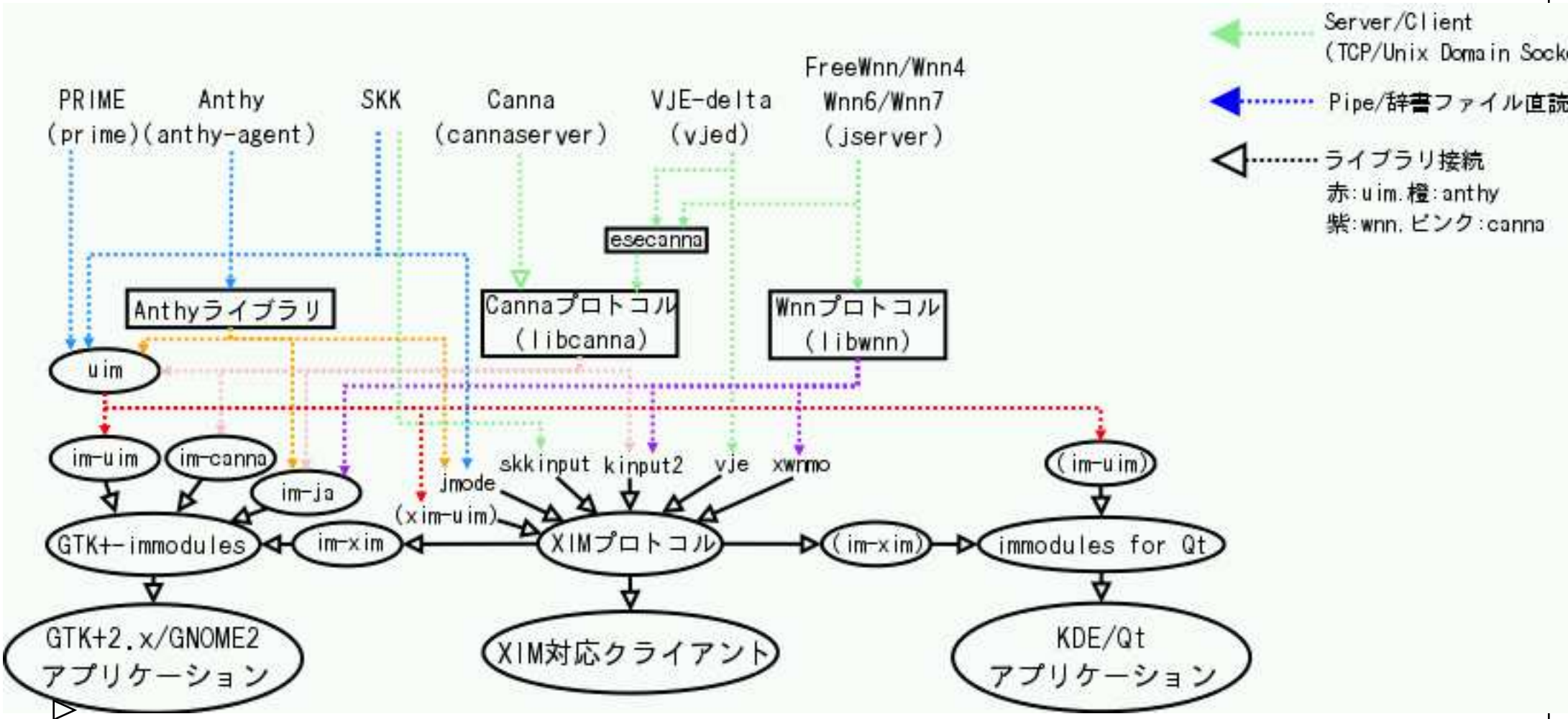
- Japanese character:
  - Hiragana/Katakana: about 100
  - Kanji: about 10,000
- Korean character:
  - EUC-KR: about 10,000
- Chinese character:
  - GB18030: over 75,000
- Taiwanese character:
  - EUC-TW: about 60,000

Japanese Input Method

- Japanese needs the most difficult/complex input system.
  - It has both phonetic (Hiragana/Katakana) and ideographic (Kanji).
  - There are many ways to input Japanese, though.
  - Analyzing input data needs translation dictionary
    - ex) Anthy has 18,000 code line + 3MB dictionary.
    - FreeWnn has 4MB dictionary.
    - Windows IME has over 60MB dictionary!
- Inputting characters all over the world can use "Finite State Model".
  - But mechanism is different between Japanese and Chinese.
- There are lots of regions to develop!

Japanese Input Method (cont)

▽image from Anthy Project



Undergoing projects

groff multibyte support

- groff 1.19 is released, but...

grep-ja, lgrep(lv)

- multibyte aware grep.
- grep can’t handle multibyte regexp.
- However, recently glibc can handle them.
- Grep should support new glibc routine, hopefully.

UTF-8 transition

- Debian Project Japanese team currently plans to support ja_JP.UTF-8.
- We plan to check "UTF-8 ready" for all locale-aware software.
  - We had experiences - transition from ja_JP.ujis to ja_JP.eucJP.
- (I plan to set) Debian JP sarge release goal as "UTF-8 ready"

UTF-8 transition (cont)

- Status - Just starting.
  - translation - mostly ok (gettext)
  - displaying - maybe ok, but some problems
  - inputing - probablly ok, but no I18N
  - printing - really bad

"Use Unicode" is not easy solution for CJK

- UTF-8 (UCS-2) is used in the software internal.
- UTF-8 is *not* used "outside" (documents/printing/displaying/...)
- We can’t convert some characters irreversible:
  - EUC-JP -> Unicode -> ShiftJIS -> Unicode -> EUC-JP
  - It becomes some problems to use iconv: ex) samba
- CJK Unicode convertion tables are a bit different between each vendors.

displaying - mojibake

- One example: debconf UTF-8

GNOME/GTK issue

- Akira TAGOH <tagoh@debian.org> is working:
  - gtk+2 uses UTF-8 in its internal.
  - You need to know what UTF-8 is.
  - Be careful to save legacy encodings.
  - Pango handles CJK font properly.
  - Use generic font name "Sans/Serif", not specific "Courier".
  - Rewrite your code with Pango if your software is gtk 1.2 based.
- Use immodule if your software is gtk 2.0 based.
- Use toolkit functions.

Resources

- The good start point is:
  - http://www.debian.org/doc/manuals/intro-i18n/

Question?