Programming for Internationalization

นายกำธร ไกรรักษ์
งานวิจัยซอฟต์แวร์พื้นฐานและทั่วไป (RD-I1)

เนื้อหา

- i18n คืออะไร?
- ความจำเป็นของ Internationalization/Localization
- Character set encoding
- Locale Data ชนิดต่างๆ
- Localization
  - การเขียนโปรแกรมเพื่อให้นำไป localize ได้
  - การ localize ซอฟต์แวร์

i18n คืออะไร?

- Internationalization (aka. i18n) คือการที่ซอฟต์แวร์สนับสนุนการนำไปใช้ในประเทศ ภาษา และวัฒนธรรมต่างๆ ที่แตกต่างกันได้ เช่น การเข้ารหัสอักขระ รูปแบบวันที่ ตัวเลข อินเทอร์เฟสภาษาต่างๆ เป็นต้น โดยเขียนโปรแกรมเพียงครั้งเดียว และใช้ไฟล์ไบนารีที่คอมไพล์แล้วตัวเดียวกันได้ทันที โดยไม่ต้องแก้ไขแล้วนำไปคอมไพล์ใหม่

ความจำเป็นของ i18n/L10N

- เพื่อให้ซอฟต์แวร์ที่พัฒนา มีความเป็นสากลมากขึ้น โดยเฉพาะซอฟต์แวร์โอเพนซอร์สซึ่งเน้นความเป็นสากล
- เพื่อความสะดวกในการนำไป Localize เป็นภาษาต่างๆ โดยไม่ต้องแก้ไขที่ซอร์สโค้ด

Thai character set encoding

- 8-bits encoding: TIS-620, ISO-8859-11, IBM-CP874, Windows-874, mac-thai
- Unicode 16-32 bits: UTF-8, UTF-16, UTF-32

Locale Data

- LC_CTYPE รหัสอักขระที่ใช้
- LC_NUMERIC รูปแบบการแสดงตัวเลข เช่น 1,200.50 หรือ 1.200,50
- LC_TIME รูปแบบการแสดงวันที่/เวลา เช่น 12 เม.ย. 2548
- LC_COLLATE การเปรียบเทียบสตริงของตัวอักษร ทำให้การเรียงลำดับถูกต้อง
- LC_MONETARY รูปแบบการแสดงตัวเลขทางการเงิน เช่น 100.00 ฿
- LC_MESSAGES ภาษาของข้อความในอินเตอร์เฟส เช่น เมนู หรือ กรอบข้อความ

Locale Data (ต่อ)

- LC_PAPER ขนาดกระดาษมาตรฐาน เช่น "A4"
- LC_NAME รูปแบบการแสดงชื่อ เช่น "บุญมี ทองมาก", “Asimov, Isaac”
- LC_ADDRESS รูปแบบการแสดงที่อยู่
- LC_TELEPHONE รูปแบบการแสดงหมายเลขโทรศัพท์ เช่น 0-2564-6900
- LC_MEASUREMENT หน่วยวัดมาตรฐาน เช่น แบบอังกฤษ (นิ้ว) หรือ ISO (เซนติเมตร)
- LC_IDENTIFICATION รูปแบบหมายเลขบัตรประจำตัว

รูปแบบของ locale

- th_TH.TIS-620
  - th = ภาษาไทย
  - TH = ประเทศไทย
  - TIS-620 = รหัสอักขระที่ใช้

การกำหนด locale

- กำหนดทั้งระบบ สำหรับ Red Hat/Fedora/LinuxTLE
  - /etc/sysconfig/i18n
    - LANG=th_TH.TIS-620
- กำหนดชั่วคราวสำหรับการรันแอพลิเคชันแต่ละครั้ง
  - $ export LC_MESSAGES=en_US
  - $ gcalctool
- หรือ
  - $ LC_MESSAGES=en_US gcalctool

การเขียนโปรแกรมให้ Localize ได้

- ใช้ฟังก์ชัน setlocale() เพื่อกำหนด locale
- ใช้ library ชื่อ gnu gettext ซึ่งเป็นมาตรฐานที่ใช้กันทั่วไปในซอฟต์แวร์โอเพนซอร์ส
- ใช้ฟังก์ชัน textdomain() เพื่อกำหนดชื่อ textdomain ที่ต้องการใช้ ซึ่งชื่อที่กำหนด จะถูกใช้เป็นชื่อไฟล์ .mo ด้วย
- ใช้ฟังก์ชัน gettext() กับ string ทั้งหมดที่ต้องการแสดงผลแบบให้สามารถนำไปแปลได้

```
/* Get setlocale() declaration.  */
#include <locale.h>

/* Get printf() declaration.  */
#include <stdio.h>

/* Get getpid() declaration.  */
#if HAVE_UNISTD_H
# include <unistd.h>
#endif

/* Get gettext(), textdomain(), bindtextdomain() declaration.  */
#include "gettext-po.h"

/* Define shortcut for gettext().  */
#define _(string) gettext (string)
#define LOCALEDIR "/usr/share/locale"

int main ()
{
  setlocale (LC_ALL, "");
  textdomain ("hello-c");
  bindtextdomain ("hello-c", LOCALEDIR);
  printf ("%s\n", _("Hello, world!"));
  printf (_("This program is running as process number %d."), getpid
());
  putchar ('\n');
  return 0;
}
```

การสร้าง และจัดการไฟล์ po

- การสร้าง template ของ po (ไฟล์ .pot)
  - `$ xgettext --keyword=_ --flag=_:1:pass-c-format -o po/hello-c.pot hello.c`
  - `$ cd po`
  - `$ cp hello-c.pot th.po`
- การแปลข้อความ สามารถใช้โปรแกรม poedit, kbabel, gtranslator เปิดไฟล์ po ขึ้นมาแปลได้
  - `$ gtranslator th.po`
- การแปลง po เป็น mo
  - `$ msgfmt -o th.mo th.po`
  - `$ cp th.mo /usr/share/locale/th/LC_MESSAGES/hello-c.mo`
- การ merge ข้อความที่แปลแล้ว กับไฟล์ pot ใหม่ เมื่อข้อความในซอร์สมีการเปลี่ยนแปลง หรือเพิ่ม
  - `$ mv th.po th.po.old`
  - `$ msgmerge th.po.old hello-c.pot > th.po`