Internacionalizace aplikací

Tématicky zaměřený vývoj aplikací v jazyce C
skupina Systémové programování – Linux

Jiří Novosad

Fakulta informatiky
Masarykova univerzita
novosad@fi.muni.cz

Brno, 15. 12. 2015

Internacionalizace aplikací

Postupy pro lokalizaci aplikací

Národní prostředí – z pohledu uživatele I

Přizpůsobení aplikací národnímu prostředí uživatele bez nunosti rekompilace.

```
man 7 locale
```

Různé kategorie

- `LC_COLLATE` – lexikografické řazení řetězců (`strcoll(3)`, `strxfrm(3)`)
- `LC_CTYPE` – třídy znaků (např. `isupper(3)`, `toupper(3)`)
- `LC_MESSAGES` – jazyk zpráv (`gettext(3)`, `rpmatch(3)`)
- `LC_MONETARY` – formát měnových řetězců (`localeconv(3)`, `strfmon(3)`)
- `LC_NUMERIC` – formát čísel (`localeconv(3)`, `printf()`, `scanf()`, ...)
- `LC_TIME` – formát času, názvů měsíců, dnů v týdnu (`strftime(3)`, `strptime(3)`)

...

Národní prostředí – z pohledu uživatele II

Proměnné prostředí

`LANG` – implicitní hodnota pro kategorie

`LC_*` – nastavení jednotlivých kategorií

`LC_ALL` – přebíjí vše

Formát locales

Jazyk – podle ISO 639-1 (cs, sk, en, . . .)

Země – podle ISO 3316 (CZ, SK, US, GB, . . .)

Kódování – ISO8859-2, UTF-8, . . .

Příklady: `cs_CZ.ISO8859-2`, `en_US.UTF-8`

Změna národního prostředí v aplikaci

```
#include <locale.h>
char *setlocale(int category, const char *locale);
```

Aplikace by při startu měla většinou zavolat:

```
setlocale(LC_ALL, "");
```

`LC_ALL` → `LC_*` → `LANG`

- Výchozí hodnota po spuštění aplikace je `"C"` nebo `"POSIX"`

úkol

- Napište program, který dostane na příkazové řádce několik řetězců.
- Řetězce vypíše uspořádané podle abecedy, bere v úvahu nastavení locale.
- Tip: `qsort(3)`, `strcoll(3)`
- `LC_COLLATE=cs_CZ.UTF-8 ./sort Cipísek čáp 123`

Internacionalizace aplikací

Internacionalizace, i18n – úprava programu tak, aby podporoval různé jazyky a kulturní zvyklosti

- podpora locale
- izolace přeložitelných dat
- infrastruktura pro překlad

Lokalizace, l10n – internacionalizovanému programu poskytneme data pro podporu konkrétního jazyka a kulturních zvyklostí (formát data a času, měna, překlad zpráv, ...)

GNU gettext()

```
#include <libintl.h>
char *gettext(const char *msgid);
char *dgettext(const char *domainname, const char *msgid);
char *dcgettext(const char *domainname, const char *msgid,
                int category);
char *bindtextdomain(const char *domainname, const char *dirname);
char *textdomain(const char *domainname);
```

- S minimálními zásahy do kódu umožňuje internacionalizaci aplikací.
- Umožňuje rozdělit práci mezi programátora a překladatele.
- Portable Object – soubor identifikátorů řetězců spolu s jejich překladem do konkrétního jazyka
- Machine Object – binární obdoba PO souborů určená pro použití aplikacemi

PO soubor – ukázka

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2010-12-12 18:08+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: hello.c:14
#, c-format
msgid "Hello world\n"
msgstr ""
```

Internacionalizace – použití I

1. Napsat aplikaci s využitím `gettext()`

   ```
   #define _(STRING) gettext(STRING)
   ```

2. Správně nastavit `locale` a doménu

   ```
   setlocale(LC_ALL, "");
   bindtextdomain("prg", "./locale");
   textdomain("prg");
   ```

3. Vytvoření generického PO souboru

   ```
   xgettext -d prg --keyword=_ prg.c
   ```

4. Vyrobení PO souboru s překladem pro konkrétní jazyk

   ```
   msginit -l cs_CZ.utf8 -i prg.po
   ```

5. Překlad a vygenerování MO souboru

   ```
   vim cs.po
   msgfmt -c -o prg.mo cs.po
   ```

6. Umístění souboru na správné místo (instalace):

   `adresar/jazyk/kategorie/domena.mo`, např.

   `./locale/cs/LC_MESSAGES/prg.mo`

úkol

- Napište Hello, world! program.
- Výchozí jazyk aplikace je angličtina.
- Připravte lokalizaci do češtiny / slovenštiny / ...

Množná čísla

```
#include <libintl.h>
char *ngettext(const char *msgid, const char *msgid_plural,
          unsigned long int n);
char *dngettext(const char *domainname, const char *msgid,
          const char *msgid_plural, unsigned long int n);
char *dcngettext(const char *domainname, const char *msgid,
          const char *msgid_plural, unsigned long int n,
          int category);
```

- Různé jazyky mají i několik forem pro množné číslo (nebo žádnou).
- PO soubor: hlavička `Plural-Forms` – udává počet forem a výraz, který pro zadané `n` vrací číslo formy
- `nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;`

úkol

- Místo Hello, world! informujte uživatele, že právě odstranil N souborů.
- N se zadává při spuštění aplikace jako parametr příkazové řádky.

Závěr

shrnutí, domácí úkoly a zdroje

Domácí úkol

Dokončete úlohy ze cvičení.

Zdroje

lokalizace a internacionalizace

- www.gnu.org/software/gettext/manual/gettext.html