

Pojęcia

Mechanizm locale

Mechanizm locale a Xlib

Kto jeszcze nie śpi?

Tłumaczenie komunikatów

catgets

Pakiet gettext

(Nawiasem mówiąc)

Nowe funkcje glibc 2.2

Problemy

Marcin Woliński

# **Narzędzia internacjonalizacji programów**

(GNU gettext/GNU libc 2.2)



*internationalization* (i18n) — operacja zastąpienia w programie rozwiązań specyficznych (najczęściej) dla angielskiego rozwiązaniami ogólnymi, przy pomocy których daje się wyrazić konwencje właściwe dla różnych narodowości i języków.  
(*internacjonalizacja?, umiędzynarodowienie?*)

*localization* (l10n) — przystosowanie programu do konwencji konkretnego języka i narodowości.  
(*lokalizacja?, unarodowienie?*)

**locale** [lou'ka:l] n. miejsce *l.* widownia  
działań *l.* wypadków.

### Kategorie locale:

LC_COLLATE	porządek sortowania
LC_CTYPE	klasyfikacja znaków (np. co jest literą, odpowiedniość małych i wielkich liter)
LC_MESSAGES	język komunikatów
LC_MONETARY	sposób wypisywania kwot pieniężnych, symbol waluty
LC_NUMERIC	sposób wypisywania liczb (np. separator dziesiętny)
LC_TIME	sposób wypisywania daty i czasu

Wyboru locale dokonuje się przy pomocy zmiennych środowiskowych:

LANG

lub dla poszczególnych kategorii: LC\_COLLATE,  
LC\_CTYPE, LC\_MESSAGES, LC\_MONETARY,  
LC\_NUMERIC, LC\_TIME

lub globalne przebicie: LC\_ALL

Na przykład:

```
export LANG=pl_PL.ISO-8859-2  
export LANG=połish
```

Powyższe są równoważne dzięki aliasom zdefiniowanym w pliku `/usr/share/locale/locale.alias`

Ustawienie zmiennych środowiskowych nie powoduje automatycznej zmiany zachowania procedur biblioteki C. W związku z tym nie każdy program jest wrażliwy na locale.

```
#include <locale.h>

main() {
    setlocale(LC_ALL, "");
    ...
}
```

```
char *setlocale(int CATEGORY, char *LOCALE)
```

Pierwszy argument `setlocale()` może być jedną ze stałych określających kategorie lub stałą `LC_ALL`. Drugi argument określa lokalizację, jaka ma być użyta; pusty napis powoduje pobranie wartości ze środowiska procesu.

Większość kategorii locale wpływa jedynie na zachowanie specjalnych, wrażliwych na locale, odmian funkcji. Na przykład do porównywania napisów przy sortowaniu zamiast `strcmp()` należy stosować `strcoll()`.

Fundamentalna różnica między programami konsolowymi a „X-owymi”:

Zwykle nieumiędzynarodowiony program konsolowy będzie w stanie czytać i wyświetlać polskie znaki, przetwarzać je będzie kulawo.

Nieumiędzynarodowiony program dla X Window nie jest w stanie odebrać polskich znaków z klawiatury.

Zdefiniowana w Xlib funkcja `XLookupString`, używana przez większość programów do dekodowania znaków przychodzących z klawiatury, jest wrażliwa na locale. Funkcja ta przynależy do kategorii `LC_CTYPE`.

Każdy program otwierający choć jedno okienko musi zawierać wywołanie `setlocale()`, żeby być używalny dla polskiego użytkownika.

```
$ bash
$ locale
LANG=POSIX
LC_CTYPE="POSIX"
LC_NUMERIC="POSIX"
LC_TIME="POSIX"
LC_COLLATE="POSIX"
LC_MONETARY="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
$ export LANG=pl_PL
$ printf "%f\n" 1.0
???
```

```
$ bash
$ locale
LANG=POSIX
LC_CTYPE="POSIX"
LC_NUMERIC="POSIX"
LC_TIME="POSIX"
LC_COLLATE="POSIX"
LC_MONETARY="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
$ export LANG=pl_PL
$ printf "%f\n" 1.0
1.000000
```

```
$ bash
$ locale
LANG=POSIX
LC_CTYPE="POSIX"
LC_NUMERIC="POSIX"
LC_TIME="POSIX"
LC_COLLATE="POSIX"
LC_MONETARY="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
$ export LANG=pl_PL
$ bash
$ printf "%f\n" 1.0
???
```

```
$ bash
$ locale
LANG=POSIX
LC_CTYPE="POSIX"
LC_NUMERIC="POSIX"
LC_TIME="POSIX"
LC_COLLATE="POSIX"
LC_MONETARY="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
$ export LANG=pl_PL
$ bash
$ printf "%f\n" 1.0
1,000000
```

Zajmiemy się bliżej kategorią locale `LC_MESSAGES`.  
Dotyczy ona języka, w jakim mają być wyświetlane komunikaty programów.

Życzymy sobie móc dodać nowe tłumaczenie bez zmian w programie (bez rekompilacji).

Dlatego zbiory komunikatów trzymane w oddzielnych plikach dla poszczególnych języków.

Procedura pobierająca tłumaczenia musi: odnaleźć i załadować plik z tłumaczeniami, na podstawie ustalonego identyfikatora odnaleźć tłumaczenie danego komunikatu.

W bibliotece GNU C dwa zestawy funkcji: `catgets` i `gettext`. Żaden nie jest zdefiniowany w standardzie POSIX. Główną różnicą jest sposób identyfikacji tłumaczonych komunikatów.

W zasadzie nie wiadomo, który mechanizm „wygra”.

W wersji catgets z każdym komunikatem musi być związany jednoznaczny identyfikator (numeryczny). Zadanie wyprodukowania jednoznacznych identyfikatorów spoczywa na programiście.

W pakiecie gettext identyfikatorem komunikatu jest tekst tego komunikatu.

Składniki pakietu:

- funkcja `gettext()` i pokrewne w bibliotece C;
- biblioteka `libintl.a` dla systemów, w których brak powyższych funkcji w bibliotece języka C;
- programy `xgettext`, `msgfmt`, `msgunfmt`, ...;
- pliki z tłumaczeniami.

Dodatkowa zmienna sterująca wyborem języka komunikatów:

```
LANGUAGE=pl:cs:de:en
```

Jej ustawienie przebija normalne ustawienia locale dla kategorii LC\_MESSAGES. Wartością może być lista języków.

```
#include <libintl.h>

main () {
    textdomain ("hello");
    puts (gettext ("Hello, world!"));
}
```

```
char *textdomain(const char *DOMAINNAME)
```

Ustanawia domyślną dziedzinę, z której należy pobierać przetłumaczone komunikaty. W istocie dziedzina jest nazwą pliku z komunikatami w katalogu `locale/<język>/LC_MESSAGES/`.

Jeśli `DOMAINNAME==NULL`, funkcja zwraca bieżącą wartość domyślnej dziedziny (nie wprowadza zmian).

```
char *gettext(const char *MSGID)
char *dgettext(const char *DOMAINNAME,
               const char *MSGID)
char *dcgettext(const char *DOMAINNAME,
                const char *MSGID, int CATEGORY)
```

DOMAINNAME==NULL znaczy „użyj domyślnej”

```
char *bindtextdomain(const char *DOMAINNAME,  
                    const char *DIRNAME)
```

Określa korzeń struktury katalogów, w której należy poszukiwać plików z komunikatami dla danej dziedziny.

### Narzędzia pakietu gettext:

- xgettext tworzy szablon tłumaczeń (plik .po) na podstawie źródeł programu
- msgfmt zamienia plik tłumaczeń na binarną postać .mo ładowaną przez gettext()
- msgunfmt dokonuje operacji odwrotnej
- msgmerge łączy dwa pliki .po (aby wykorzystać tłumaczenia wykonane dla starszej wersji programu)

Wynik uruchomienia programu xgettext na pliku hello.c (fragment):

```
#: hello.c:4  
msgid "Hello world!\n"  
msgstr ""
```

Po interwencji tłumacza (plik pl.po):

```
#: hello.c:4  
msgid "Hello world!\n"  
msgstr "Witaj świecie!\n"
```

[http://www.li18nux.org  
/root/LI18NUX2000/LI18NUX-2000.pdf](http://www.li18nux.org/root/LI18NUX2000/LI18NUX-2000.pdf)

LI18NUX 2000  
Globalization Specification Version 1.0  
Linux Internationalization Initiative (Li18nux)

Copyright 2000 The Free Standards Group.

```
printf("%d file(s) deleted.\n", delnum);
```

```
printf(gettext("%d file(s) deleted.\n"),  
       delnum);
```

```
msgid "%d file(s) deleted.\n"  
msgstr "Skasowano %d plik(ów).\n"
```

Skasowano 2 plik(ów).

```
char *ngettext(  
    const char *MSGID1, const char *MSGID2,  
    unsigned long int N)
```

```
char *dngettext(const char *DOMAIN,  
    const char *MSGID1, const char *MSGID2,  
    unsigned long int N)
```

```
char *dcngettext(const char *DOMAIN,  
    const char *MSGID1, const char *MSGID2,  
    unsigned long int N, int CATEGORY)
```

W pliku PO:

```
msgid ""  
msgstr "nplurals=3; plural= n==1 ? 0 : \  
n%10>=2 && n%10<=4 && \  
(n%100<10 || n%100>=20) ? 1 : 2"
```

```
msgid "%d file deleted.\n"  
msgstr0 "Skasowano %d plik.\n"  
msgstr1 "Skasowano %d pliki.\n"  
msgstr2 "Skasowano %d plików.\n"
```

węgierski, japoński, turecki: `nplurals=1; plural=0`

angielski, niemiecki, ...: `nplurals=2; plural=n!=1`

francuski: `nplurals=2; plural=n>1`

rosyjski: `nplurals=3; plural=n%10==1 ? 0 :`

`n%10>=2 && n%10<=4 &&`

`(n%100<10 || n%100>=20) ? 1 : 2`

polski: `nplurals=3; plural=n==1 ? 0 :`

`n%10>=2 && n%10<=4 &&`

`(n%100<10 || n%100>=20) ? 1 : 2`

słoweński: `nplurals=4; plural=n==1 ? 0 :`

`n%10==2 ? 1 : n%10==3 || n%10==4 ? 2 : 3`

```
printf(ngettext("%d file deleted.\n",  
                "%d files deleted.\n", delnum),  
       delnum);
```

```
msgid "%d file deleted.\n"  
msgstr0 "Skasowano %d plik.\n"  
msgstr1 "Skasowano %d pliki.\n"  
msgstr2 "Skasowano %d plików.\n"
```

Skasowano 2 pliki.

```
printf("You have %d %s and %d %s in your pocket.\n",  
      appnum,  
      ngettext("apple", "apples", appnum),  
      sheepnum,  
      ngettext("sheep", "sheep", sheepnum) );
```

```
msgid "You have %d %s and %d %s in your pocket.\n"  
msgstr "Masz w kieszeni %d %s i %d %s.\n"
```

```
msgid "sheep"  
msgstr0 "owcę"  
msgstr1 "owce"  
msgstr2 "owiec"
```