Többnyelvű programok létrehozása a GNU gettext rendszerrel

A legtöbb kereskedelmi és ingyenes programot manapság angol nyelven írják és dokumentálják. Csak az utóbbi időben tettek kísérletet arra, hogy ezen kívül más nyelven is próbáljon a számítógép kommunikálni a felhasználóval. A GNU gettext program megjelenése sokat javított a helyzeten.

A legtöbb *GNU* program ma már a *gettext* kerettel készül, amely lehetővé teszi a programok egyszerű fordítását. Ebben a rendszerben, ha rendelkezésre áll a fordítás, akkor a programoknál futás előtt, vagy akár közben állíthatjuk a nyelvet. A *gettext* egy olyan egyszerű módon oldja meg ezt, amely megkönnyíti a programozó és a fordítók munkáját is, mi több, külön, egymástól függetlenül tudnak dolgozni.
Ez az írás a *gettext* rendszert mutatja be. Létezik egy másik hasonló célú programcsomag is, a *catgets*, amelyik az *X/Open Portability Guide* ajánlása, de azt nem részletezzük.

Az első program

Kezdjük a *gettext* megismerését is a szokásos *„Hello, world”* programmal! Ennek a nemzetközi változata a következőképpen néz ki:

```
#include <libintl.h>
#include <locale.h>
#include <stdio.h>
int main()
{
  setlocale(LC_ALL, "");
  bindtextdomain
  ➥("hello", "/usr/
  ➥share/locale");
  textdomain("hello");
  printf(gettext("Hello,
  ➥world!\n"));  return 0;
}
```

Nyilván egy igazi programnál érdemes a visszatérési értékeket ellenőrizni. Mentsük el ezt a kis forráskódot *hello.c* néven, és fordítsuk le a programot a szokásos

```
gcc -o hello hello.c
```

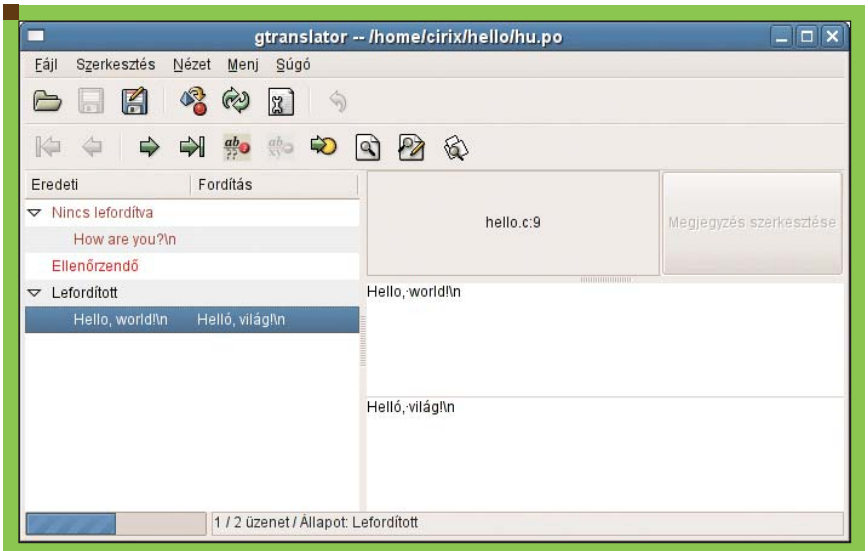
paranccsal. A programhoz kell szerkeszteni a *GNU libintl* könyvtárat; *GNU* rendszerekben ezt a *glibc* tartalmazza, máshol szükség lehet a `-lintl` paraméterre is.

A programozó feladata

A *hello* program a szokásos „Hello, world!” üzenetet írja a képernyőre. Néhány függvényhívás után, amelyeket csak a program indulásakor kell megtenni, az összes feladata a programozónak, hogy minden karakterláncot a `gettext(string)` függvénnyel feldolgoztasson kiíratás előtt. Megszokott ennek a rövidítése is, hogy még kevesebbet kelljen gépelni:

```
#define _(STRING)
➥gettext(STRING)
...
printf(_("Hello, world!"));
```

Nézzük a programot soronként. A *locale.h* néhány területi információt tároló C struktúrát határoz meg. Ez tartalmazza a `setlocale()` függvény prototípusát. A `libintl.h` pedig a *GNU gettext* függvényeket tartalmazza, a `bindtextdomain()`-t, a `gettext()`-et és a `textdomain()`-t. A `setlocale(LC_ALL,"")` függvényhívás beállítja a program területi paramétereit a környezeti változók, vagyis a felhasználó kérése alapján. A `bindtextdomain()` függvény megadja, hogy egy adott nyelv üzenet fájljai melyik könyvtárban találhatóak. A csomag neve általában megegyezik a program nevével.



1. ábra A gtranslator munka közben

Az üzenetkatalógusok alapértelmezés szerinti helye az */usr/share/locale* könyvtár (ebben az esetben egyébként el is hagyhattuk volna ezt a függvényhívást).
A `textdomain()` függvényhívás kiválasztja a ***hello*** nevű üzenetkatalógust.
A `printf("Hello, world\n");` sort pedig kicseréltük a `printf(gettext("Hello, world!\n"));` sorra. Ez a módosítás teszi lehetővé, hogy a fordító a programozótól külön dolgozhasson. A paramétere az angol nyelvű mondat. Visszatérési értéke pedig egy olyan karakterláncra mutat, amely a memóriában az adott nyelvre lefordított szöveget tartalmazza. Ha nem talál fordítást, az eredeti mondattal tér vissza. (Ezért látni a frissebb programokban elszórva angol szavakat.)

A fordítandó üzenetek

Kerestessük ki a fordítandó üzeneteket a forráskódból. Erre az `xgettext` program való, amelyet a következő paranccsal kell elindítanunk:

```
xgettext -d hello -s -o
➥hello.pot hello.c
```

Ez feldolgozza a ***hello.c*** forrást, és elkészíti a ***hello.pot*** fájlt (amelyet a `-o` paraméterrel adtunk meg, output). A `-s` hatására (sort) ABC szerint rendezi az üzeneteket a program, a `-d` pedig az üzenetkatalógus nevét (domain) állítja be, amelyet a `textdomain()` függvényhívásnak is meg kell adnunk.
A ***hello.pot*** (portable object template) fájl egy sablon, amely alapja a fordításokat tartalmazó fájloknak. A fordítás elkezdéséhez lemásolhatjuk a sablon fájlt egy másik néven, de a egyszerűbb módja ennek az ***msginit*** program használata, amely néhány változót egyből beállít a helyes értékre:

```
msginit -l hu_HU -o hu.po -i
➥hello.pot
```

Itt a `-l` paraméter (locale) adja meg a terület és a nyelv nevét, a `-i` és a `-o` paraméterek pedig a bemeneti és a kimeneti fájlokat (input, output). Ha csak egy sablon fájl van az aktuális könyvtárban, azt használja a program automatikusan. Az ***msginit*** megkérdezi a fordító e-mail címét.
A ***hu.po*** (portable object) fájlban az egyes bejegyzések formátuma a következő:

```
#  fordító megjegyzései
#. automatikus megjegyzései
#: hivatkozás
#, flag-ek
msgid "fordítatlan üzenet"
msgstr "lefordított üzenet"
```

A megjegyzések a `#` (kettőskereszt) karakterrel kezdődnek. Két fajtájuk van. Az első a kézzel, fordító által hozzáadott megjegyzések, amelyeknél a kettőskeresztet egy szóköz követi. A második pedig automatikus megjegyzés a ***gettext*** program által. Az `msgid` után a fordítatlan (általában angol nyelvű) üzenet van, az `msgstr` után pedig a lefordított üzenetet kell beírni.

A fordítás

A ***hu.po*** fájlt szerkeszthetjük a fordításhoz. Erre bármilyen szövegszerkesztő program alkalmas, ha az ember pontosan betartja a formátumot; érdemesebb azonban egy erre a célra való programot használni. Ilyenek a ***kbabel***, a ***gtranslator***, a ***poedit***; az ***Emacs*** is képes ***.po*** módban dolgozni.
Először a fejlécet töltsük ki. Ezt részben az ***msginit*** program elvégezte. A `Project-Id-Version` után írjunk annyit, hogy `hello 0.1`. A munka nagy része az egyes üzenetek fordítása. Minden `msgid` sor után írjuk be az `msgstr`-hez a magyar szöveget, idézőjelek között. Vagyis az msgid `"Hello, world!\n"` után `msgstr "Helló, világ!\n"` kerül. Figyeljünk arra, ha elmentjük a fájlt, a `Content-Type` sornál megadott karakterkészletet használjuk. Magyar nyelv esetén ez ***ISO-8859-2*** vagy ***UTF-8*** lehet. Az ***msginit*** program a kiválasztott nyelvkódtól függően állítja ezt be.

Az üzenetkatalógusok

A fordítás elvégzése után a ***.po*** szövegfájlt bináris formátumra kell hoznunk, hogy abban a ***gettext*** programok gyorsan meg tudják találni a karakterláncokat. Ehhez a következő parancsot kell használni:

```
msgfmt -c -v -o hello.mo hu.po
```

A `-c` opció részletesen ellenőrzi a ***.po*** fájlt. A `-v` hatására részletes üzeneteket kapunk. A kimeneti fájl nevét a `-o` opció után adhatjuk meg. Fontos, hogy a kimeneti fájl neve az üzenetkatalógus nevével egyezzen meg, amelyet a `bindtextdomain()` és `textdomain()` függvénynek is megadunk. A ***.mo*** (machine object) fájl a `bindtextdomain()` második paraméterében megadott könyvtárban kell legyen a program futásakor. A teljes elérési út a ***megadott_könyvtár/nn_OO/LC_MESSAGES/hello.mo***,

ahol nn a nyelv (hu) és OO az ország (HU). Így különböztetik meg például a kanadai franciát a franciaországitól: fr_CA és fr_FR. A *hello.mo* fájl szabványos helyre másolásához természetesen rendszergazdai jogosultság szükséges:

```
mkdir -p /usr/share/locale/
➥hu_HU/LC_MESSAGES
cp hello.mo /usr/share/locale/
➥hu_HU/LC_MESSAGES
```

(Ha enélkül szeretnénk kipróbálni a *gettext*-et, legyen a `bindtextdomain()` második argumentuma "" üres karakterlánc, de a *hello.mo* fájlt ebben az esetben is az aktuális könyvtárhoz képest a *hu_HU/LC_MESSAGES* helyre kell másolnunk!)

A felhasználó feladata

Ha az üzenetkatalógusokat megfelelően telepítettük, a rendszer bármelyik felhasználója kipróbálhatja a magyar nyelvű *Helló, világ* programot. Állítsuk be a megfelelő környezeti változót (bár ezt valószínűleg a *Linux* terjesztésünk már tartalmazza), utána pedig indítsuk el a programot:

```
export LC_ALL=hu_HU
./hello
```

Figyeljünk arra, hogy terminálunk vagy terminál ablakunk ismerje a megfelelő nem angol nyelvű betűket, különben a kimenet hibás lehet.

A program továbbfejlesztése – új üzenetek

Bővítsük kicsit az előbbi programunkat. Adjunk hozzá még egy mondatot:

```
...
printf(gettext("Hello,
➥world!\n");
printf(gettext("What's up?\n");
...
```

Ha csak ennyit változtatunk, legegyszerűbb lenne újra végigcsinálni a fenti lépéseket, és lefordítani mind a két üzenetet magyar nyelvűre. Egy nagy, több ezer üzenetet tartalmazó programnál nem ilyen egyszerű a helyzet. Jó lenne, ha csak az újonnan megjelent sorokat kellene újra lefordítanunk.

Keressük ki az *xgettext* programmal újra a fordítandó mondatokat:

```
xgettext -d hello -s -o
➥hello_uj.pot hello.c
```

Az *msgmerge* program egy régebbi, már lefordított *.po* fájlt tud összefűzni egy új .pot (sablon) fájllal:

```
msgmerge -s -U hu.po
➥hello_uj.pot
```

A `-U` opcióval (update) jelezzük, hogy a régi *hu.po* fájlt szeretnénk továbbfejleszteni. Az új sablon fájl tartalmazni fogja a régi fordításokat, az újonnan megjelent soroknál pedig üres karakterláncot. Fordítsuk le az új sort:

```
#. hello.c:11
msgid "What's up?\n"
msgstr "Mi a helyzet?"
```

Ha valamelyik *.po* fájl szerkesztő programot használjuk, akkor az külön mutatja a már lefordított és még feldolgozandó sorokat. Ezután pedig készítsük el a *.mo* fájlt, ahogy a fenti esetben:

```
msgfmt -c -v -o hello.mo hu.po
mkdir -p /usr/share/locale/
➥hu_HU/LC_MESSAGES
cp hello.mo /usr/share/locale/
➥hu_HU/LC_MESSAGES
```

Egyéb tudnivalók

Néhány dologra figyelnünk kell a fordítás közben. Legfontosabb, hogy ha a fordítandó üzenetünk C formátum karaktereket tartalmaz, azokat meg kell tartanunk, méghozzá a megfelelő sorrendben. Megfelelően fogalmazva a mondatot ez könnyen elérhető. Például:

```
#  helyes
msgid "Read %d lines from file
➥%s.\n"
msgstr "%d sor beolvasva a %s
➥fájlból.\n"
```

Helytelen viszont a következő sor:

```
#  helytelen
msgstr "%s fájlból beolvasva
➥%d sor.\n"
```

A C formátum karakterláncokat az *xgettext* megjelöli a `#, c-format` megjegyzéssel. (Ennek ellenkezője a `#, no-c-format`). A `fuzzy` jelzőt is ennél a sornál állíthatjuk be vagy törölhetjük. „Fuzzy" jelölést szokás tenni az ellenőrzendő vagy pontatlan fordítások mellé. Ezeket a szerkesztő programok külön kategóriába teszik.
Grafikus rendszerek gyakran _ (aláhúzás) karakterrel jelölik a gyorsbillentyűket. Ezen sorok fordítását fontolóra kell venni. Figyelni kell rá, hogy ne adjuk több gombnak vagy menünek ugyanazt a gyorsbillentyűt.
Az *xgettext* program a `-c` opcióval indítva az egyes karakterláncok elé megjegyzést is ír, amelyet a forráskódból másol át. Minden `msgid` elé az a megjegyzés kerül, amely a kódban a `gettext()` függvényhívást megelőzte. Ha ilyeneket helyezünk el a programunkban, az nagyban megkönnyítheti a fordító munkáját. Fontos azonban, hogy ez a megjegyzés csak *ASCII* karaktereket tartalmazzon, vagy adjuk meg azok karakterkódolását is a `--from-code` opcióval! Ellenkező esetben az *xgettext* nem fut le. Ráadásul a jelenlegi verzió (0.14.5) a `gettext()` függvénynek átadott karakterláncot jelzi hibásnak, nem pedig a megjegyzést!
Az *msgmerge* program #~ jelzéssel látja el azokat a sorokat, amelyek egy új programverzióban már nem szerepelnek. Ha nem a saját programunkat fordítjuk, érdemes ezeket figyelmen kívül hagyni; semmiképpen sem kitörölni, hátha egy későbbi verzióban megint szerepel majd.
A *gettext* csomag része egyébként a gettextize program is, amely felkészít egy programrendszert a *gettext* használatára, vagy egy újabb *gettext* verzióra. Ennek és az eddig említett programoknak a használatát a ➲ *http://www.gnu.org/software/gettext/manual/* oldal mutatja be.

Czirkos Zoltán
Jelenleg diplomatervező a Budapesti Műszaki Egyetem Elektronikus Eszközök Tanszékén. Kutatási területe az operációs rendszerek betörésvédelme és a P2P kommunikáció. 2005-ben a Tudományos Diákköri Konferencián II. helyezést ért el. Kedvencei a boszorkányos és a rózsaszín párducos filmek.