Localizzazione dei programmi - l10n

[Questo articolo analizza il procedimento di internazionalizzazione e quello di localizzazione di un programma dal punto di vista del traduttore (o del sistemista), più che da quello del programmatore. Spero che possa essere uno spunto valido per chi ha il desiderio di contribuire allo sviluppo del software open source senza essere un programmatore.]

Recentemente mi sono imbattuto per ben due volte, a distanza ravvicinata di tempo, in problemi legati alla traduzione di programmi.
Il primo caso è legato alla necessità di tradurre un mio programma e quindi alla parte di programmazione del processo di traduzione. Appena avrò risolto completamente i problemi relativi, cercherò di scrivere qualcosa a questo proposito.
Il secondo è legato alla mia traduzione del programma Gaby, un programma per la gestione di piccoli database al quale vorrei dedicare il prossimo articolo. Da quest'ultima esperienza ho rilevato come, nonostante le mie buone conoscenze dell'argomento, io abbia incontrato dei problemi nell'affrontare la traduzione. Ho deciso quindi di scrivere questi appunti, tentando una semplificazione dell' ampia documentazione esistente, nella speranza che possano essere utili a qualcuno.

Come, probabilmente, molti di voi ben sanno, la maggior parte dei programmi (open source e non) vengono sviluppati in lingua inglese. Molti utenti non si sentono sufficientemente a loro agio nell'utilizzare tale lingua. I programmatori hanno quindi dovuto, da sempre, dedicare parte delle loro energie all' internazionalizzazione e localizzazione dei programmi, rispettivamente (per gli amanti delle sigle) i18n e l10n.
Per chi non conoscesse la differenza fra i due termini dirò, semplificando, che la conversione di un programma atta a farlo diventare multilingua è comunemente detta globalizzazione. La globalizzazione di un programma passa attraverso due fasi.
La prima è il processo di internazionalizzazione che consiste nella revisione del codice di un programma in modo da renderlo atto ad essere "localizzato". Questa è una procedura generica che vale per qualsiasi numero di lingue il programma debba supportare; essa deve tenere conto della parte più tecnica del processo e di problematiche relative ai formati di data, di ora, dei numeri reali, delle lettere accentate, dei segni di interpuntazione e della direzione della scrittura (in alcune lingue come l'arabo o il giapponese si scrive da destra a sinistra o addirittura dal basso in alto) e similari.
La seconda fase, detta localizzazione, è, invece, la traduzione del programma in lingue diverse da quella nella quale è stato scritto. Non è forse così ovvio che, tale fase è forse la più onerosa in quanto va affrontata programma per programma ed è soggetta a revisione continua all'evolversi dello stesso. Oltretutto è la parte su cui gli automatismi meno riescono a essere efficaci come ben sa chiunque di voi abbia visto una traduzione automatica di un testo.

Pur non volendo approfondire troppo il lato tecnico è opportuno vedere quali sono i passi per la produzione di un programma multilingua ovvero individuare i punti nei quali un traduttore può intervenire in maniera autonoma.
Dovete sapere che un primitivo metodo di globalizzazione consisteva nel fork (divisione) dello sviluppo di un programma, attraverso il quale un nuovo team di programmatori si prendeva carico di creare una versione tradotta dei programmi. Tale metodo è ovviamente oneroso e crea ritardi nello sviluppo delle versioni localizzate dei programmi oltre a portare bug aggiuntivi negli stessi.
Chiaramente la situazione si è evoluta ed oggi quasi tutti i linguaggi di programmazione permettono l'internazionalizzazione, (d'ora in poi i18n) tramite l'uso di apposite librerie. Su tali argomenti vorrei soffermarmi in un futuro articolo (come accennato in apertura) dedicato alla parte più tecnica dell'i18n e alla programmazione.
Per il momento sia sufficiente sapere che le funzioni più utilizzate sono la standard X/Open P.G. `catgets()` e, in ambiente GNU/Linux, la funzione `gettext()` che portano a corredo dei tools per automatizzare alcune operazioni di manutenzione.
Tramite tali funzioni il programma va a "leggere" la scritta da visualizzare su un file esterno al programma.
Nel caso della funzione gettext, la struttura di tale file è molto semplice, si tratta di un file di "hash", ovvero una sorta di tabella, nella quale vengono elencate le scritte originali e di seguito la corrispondente traduzione. La scritta originale viene, quindi, utilizzata come identificativo tramite il quale viene effettuata la ricerca della frase tradotta all'interno del suddetto file di traduzione.
Il file in realtà non è unico: ne esiste uno per ogni linguaggio nel quale il programma è stato tradotto. Ciascun file ha un nome di due lettere che corrispondono allo stato/lingua al quale esso si riferisce. Tali lettere sono costituite dal codice internazionale del paese che per l'Italia è it.
La selezione del file corretto avviene durante l'esecuzione del programma basandosi su una variabile d'ambiente di nome LOCALE che ha valore uguale al codice internazionale dello stato. Chiaramente il presupposto per un corretto funzionamento è che il file con le scritte "it" da visualizzare esista e sia correttamente formato.

Ma andiamo con ordine:

Partirò dal presupposto che il programma che vi troverete di fronte sia già passato per il processo di i18n e che tale processo sia stato operato mediante l'uso della funzione gettext() che è la più diffusa ed utilizzata.

a) il programmatore ha preparato correttamente il programma **prova** per l'i18n (scusate la scarsa originalità nel nome del programma).

b) egli genera tramite il programma `xgettext` il file `prova.pot`. Tale file, a quel punto, contiene tutte le frasi in lingua originale (es. inglese). Ogni frase è racchiusa tra apici ed è preceduta dalla dicitura **msgid**. Sotto la frase compare la dicitura **msgstr** e una coppia di apici. Tale dicitura serve da marcaposto per la traduzione.

L'esempio che segue forse vale più di mille parole (è tratto dalla traduzione di Gaby di cui sopra):

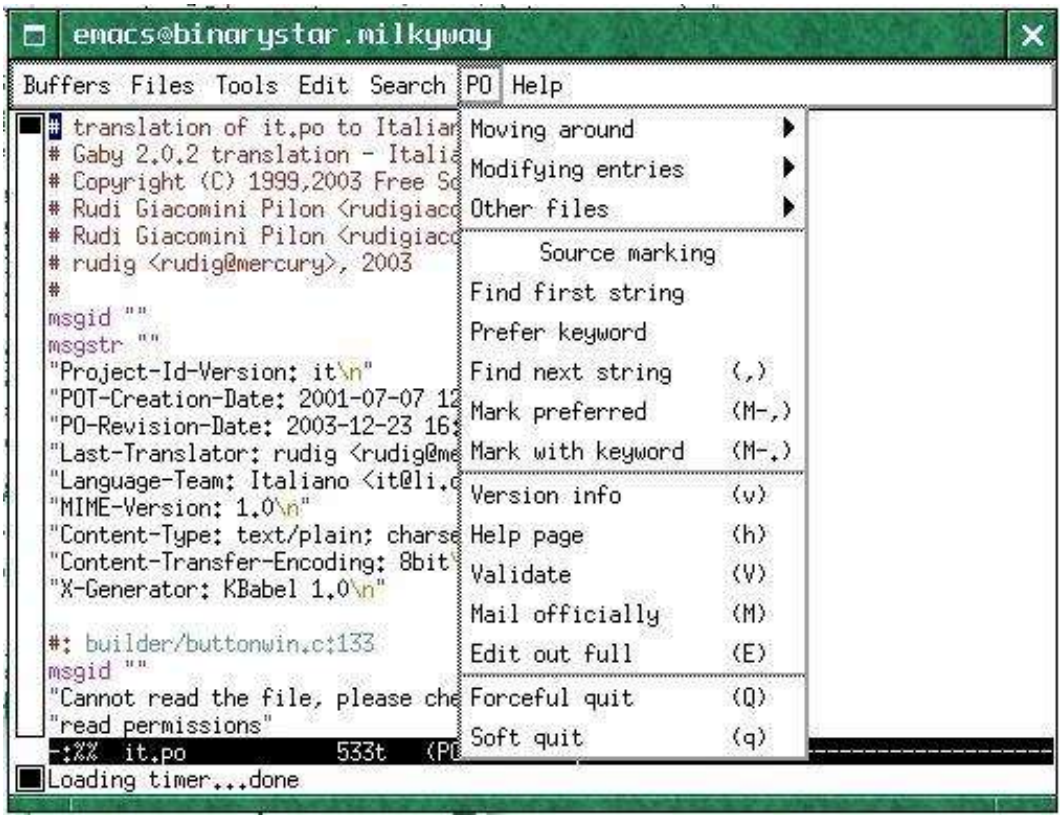
```
#: builder/buttonwin.c:605 builder/fileops.c:317
msgid "Open"
msgstr ""

#: builder/buttonwin.c:605
msgid "Opens a file"
msgstr ""
```

c) a questo punto tocca a voi, qualora abbiate scelto il ruolo del traduttore. Innanzitutto è opportuno contattare, con una mail, il responsabile dello sviluppo per chiedere se sia già in corso qualche porting del programma nella vostra lingua. In caso affermativo potete unirvi al team di traduzione che il responsabile vi indicherà, in modo da non sprecare i vostri sforzi inutilmente con un lavoro duplicato.
In caso negativo state per diventare il responsabile per la traduzione nella vostra lingua e potete iniziare ad operare la traduzione.

d) dovrete quindi copiare il file `prova.pot` in `it.po` (nel caso dell'italiano) ovvero in un file, con estensione `.po` il cui nome di due lettere è la sigla della nazione nella quale lingua si vuole tradurre.

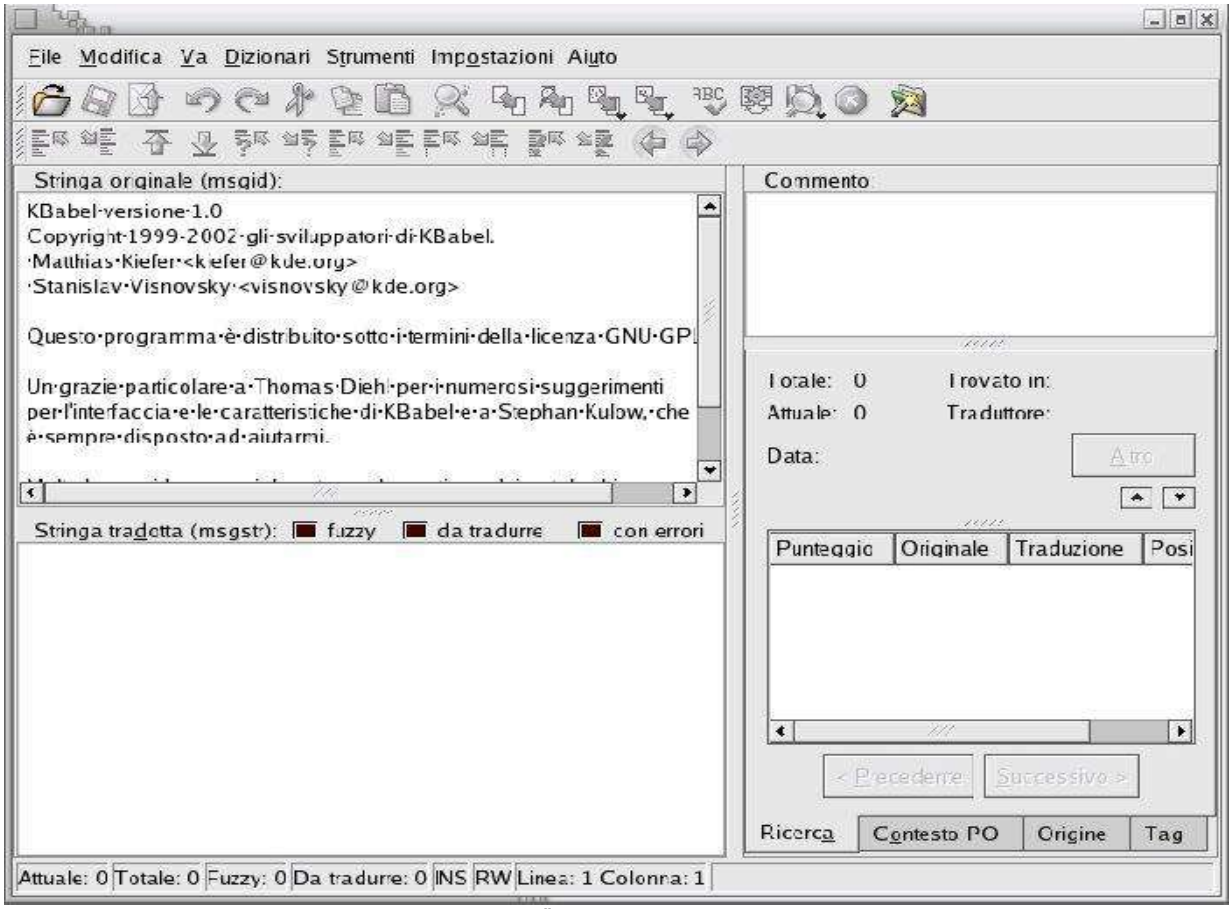
e) potete ora aprire il file con il vostro editor preferito, per le modifiche, tenendo conto che esistono due strumenti più adatti di altri per la traduzione.
Il primo, e più tradizionale, è emacs (ovviamente). Esiste una apposita modalità operativa di emacs tramite la quale si può disporre di menù e macro ottimizzate per lavorare con i files .po. Per attivare il PO-mode potete modificare il file .emacs come segue:

```
(setq auto-mode-alist
 (cons '("\\.po\\'\\|\\.po\\." . po-mode) auto-mode-alist))
 (autoload 'po-mode "po-mode" "Major mode for translators to edit PO files" t)
```



Tale modalità attiva un menù/una modalità comandi aggiuntiva che facilita alcune operazioni. Dovete sapere che le frasi possono avere tre stati diversi: si considerano da tradurre quando il campo msgstr è vuoto, tradotte quando il campo msgstr è non-vuoto, "fuzzy" quando il campo msgstr è non-vuoto ma la traduzione è non corretta o obsoleta. Tramite il po-mode, per esempio, vengono facilitati lo spostamento alla successiva frase da tradurre tramite il tasto **t** o alla precedente tramite il tasto **T**, lo spostamento alla successiva frase contenente un errore con il tasto **z** o alla precedente tramite **Z**, o ancora la validazione (V), l'invio delle traduzioni e molte altre operazioni. A chi volesse dei dettagli consiglio la lettura della sezione relativa al po-mode all'interno del manuale di gettext().

Il secondo strumento è kbabel che, essendo uno strumento nato per questo uso è, forse, il sistema ottimale per la traduzione.



"

Qui le operazioni sono ulteriormente facilitate dall'interfaccia grafica e lo spostamento fra una frase è l'altra avviene attraverso dei comodi bottoni.
Inoltre la visualizzazione delle frasi è ottimizzata: in alto a sinistra avete la frase originale da tradurre (msgid) in basso a sinistra avete lo spazio per la frase tradotta (msgstr) a destra la visualizzazione dei commenti relativi alla frase e di una serie di informazioni aggiuntive.

f) come già detto le varie frasi vengono tradotte all'interno di msgstr senza alterare il msgid e rispettando eventuali variabili.
L'esempio che segue riprende le righe sopra presentate dopo l'aggiunta delle frasi tradotte:

```
#: builder/buttonwin.c:605 builder/fileops.c:317
msgid "Open"
msgstr "Apri"

#: builder/buttonwin.c:605
msgid "Opens a file"
msgstr "Apre un file"
```

chiaramente se la frase da tradurre contiene delle variabili o dei caratteri speciali essi vanno mantenuti e posizionati opportunamente come nell'esempio che segue

```
#: src/actions.c:77
#, c-format
msgid "Unable to find %s\n"
msgstr "Impossibile trovare %s\n"
```

dove `%s` rappresenta una variabile di stringa e `\n` un ritorno a capo.
Le frasi che iniziano con **#:** sono dei riferimenti al sorgente, quelle con **#;** degli attributi delle frasi da tradurre, il solo simbolo **#** all' inizio di una riga, invece, indica un commento.

g) dal file it.po tramite il comando:
`msgfmt it.po -o it.gmo`
viene creato il file formattato che poi sarà linkato al programma in fase di compilazione.

h) se non avete competenze di programmazione il vostro compito è quasi completato. Non vi resta infatti che inviare al responsabile dello sviluppo del software i files contenenti le traduzioni in modo che vengano incorporati nella prossima release.

Se invece avete un minimo di conoscenze di base potete fare ancora qualcosa testando la traduzione nel vostro PC.

i) per fare questo sarà necessario modificare alcuni files. Se i sorgenti sono stati preparati secondo le specifiche indicate nel manuale di gettext, portandovi nella root dei sorgenti del programma, dovreste trovarvi di fronte ad una struttura ad albero nella quale sono presenti almeno la directory /po, /intl e /src.
La prima è il vostro obbiettivo ed è all'interno di essa che dovrete inserire i files it.po e it.gmo. In tale directory dovrebbe essere inoltre presente il file `LINGUAS`. Apritelo con un editor e aggiungete `it` all'elenco dei linguaggi disponibili.

l) dovrete poi effettuare la modifica del file `configure` contenuto nella root dei sorgenti. Individuate una riga simile alla seguente:

```
ALL_LINGUAS="da de es fi fr ja nl no pl sv"
```

e aggiungete it fra le lingue disponibili.

m) effettuate la ricompilazione del programma con i comandi `configure` e `make`

n) a questo punto la magia. Settando da terminale la variabile d'ambiente: `export LANGUAGE="it"` e lanciando il programma **prova** (dopo averlo compilato) vedremo le scritte in italiano.

o) verificate bene il posizionamento delle varie diciture, che nessuna di esse risulti troncata o visualizzata in maniera impropria. Verificate anche la coerenza della scritta con l'azione a cui si riferisce. Se tutto vi sembra corretto potete inviare i files tradotti al manutentore del programma.

A questo punto un paio di suggerimenti:
Se volete intervenire nello sviluppo di un programma con la traduzione dovreste scegliere il momento opportuno per farlo. Se il programma è immaturo o in una forte fase evolutiva c'è il rischio che dobbiate intervenire molto spesso nella traduzione. Per contro evitate di affrontare la traduzione di un programma obsoleto e destinato a scomparire.
Attenzione, inoltre, alle trappole linguistiche dovute al "linguaggio informatico". La traduzione di un programma non consiste nella conversione, parola per parola, dei vocaboli, da una lingua all'altra, o nella traduzione più o meno letterale delle frasi. Anni fa, per esempio, una nota multinazionale informatica (evitiamo nomi e cognomi) traduceva, nei manuali, termini come hard-disk e mouse rispettivamente in "disco duro" e "topo" che suonavano a dir poco ridicoli. In ambito informatico, spesso, molte espressioni sono gergali e non mantengono nessun riferimento alla voce originale.

Riferimenti

Riferimenti alla funzione catgets
Manuale di gettext
Home page di Kbabel

di Rudi Giacomini Pilon