Localisation, Internationalisation

S. Aicardi

ANF Mathrice, Angers, 21-25 mai 2012

Un ordinateur polyglotte ?

L'informatique a été essentiellement développée par et pour des américains. L'anglais est donc la langue privilégiée des programmeurs.

La plupart des systèmes et programmes que nous utilisons aujourd'hui descendent plus ou moins directement de systèmes mis au point dans les années 1960-1970, à une époque où la faible puissance et capacité des machines contraignait les systèmes à une économie drastique de moyens : peu d'espace de stockage, pas d'écran, imprimantes rudimentaires.

Un ordinateur polyglotte ?

Cela implique des présupposés :

- on écrit de gauche à droite et de haut en bas,
- on utilise l'alphabet latin
- on peut tout écrire avec 26 lettres, 10 chiffres et 33 signes de ponctuation,
- une lettre s'écrit toujours de la même manière quel que soit le contexte,
- la monnaie est le dollar,
- la virgule sert à séparer les milliers, le point à séparer la partie décimale,
- etc.

Top 10 des langues maternelles

Langue	Locuteurs	Exemple
Mandarin	849 M	世界你好！
Espagnol	329 M	¡Hola, mundo!
Anglais	328 M	Hello, world!
Arabe	221 M	مرحبا العالم
Hindi	182 M	नमस्ते दु निया
Bengalî	181 M	নমস্কার৷
Portugais	178 M	Olá, mundo!
Russe	144 M	Здравствуй, мир!
Japonais	122 M	世界よ、こんにちは!
Allemand	90 M	Grüße!
…	…	
Français	70 M	Bonjour à tous !

Source

Ubiquité de l'informatique et d'Internet

Dans les années 1980-1990, l'informatique se diffuse dans le monde professionnel, puis personnel. Le réseau mondial interconnecte le monde entier. Aujourd'hui, Internet connecte 35 % de la population mondiale. Il y a déjà plus de chinois que d'américains sur Internet et le chinois est en passe de sépasser l'anglais comme langue d'usage.

Un programme doit prévoir son utilisation par tout internaute, quelque soit sa langue ou sa culture.

Suffit-il de traduire ?

L'objectif est de rendre l'interface utilisateur aussi naturelle que possible à un non-anglophone.

Il faut donc traduire en respectant à la fois la langue et l'interface utilisateur : vocabulaire concis et clair, syntaxe, ponctuation, expressions idiomatiques.

Le plus difficile est souvent de trouver de bons traducteurs pour la plupart des langues !

Au-delà des problèmes de langue

De multiples conventions d'écriture :

- Nombres : 1,234.56 ? 1 234,56 ? MCCXXXIV ? ,αση´ ? غرلد ?
- Monnaie : $50 ou 50 € ?
- Dates : 9/11/2001 ou 11/9/2001 ?
- Unités : in ou cm ? oz ou g ? BTU ou J ? letter ou A4 ? Farenheit ou Celcius ? mph ou km/h ?
- Heure : 1 PM PST ou 13:00 CEST ?

Problèmes de codage

Dès l'invention des télécommunications, il a fallu résoudre le problème du codage des caractères. Du sémaphore au télégraphe, les standards locaux, nationaux et internationaux se multiplient. En 1874, le code Baudot sur 5 bits est le premier code binaire.

Aux débuts de l'informatique, l'information est stockée sur 6 bits, puis rapidement sur 7 ou 8 bits.

Mais les échanges internationaux de données sont rares et les codages nationaux se multiplient.

Un long processus de standardisation : ASCII

Le codage ASCII sur 7 bits apparaît en 1963 et standardisé en ISO-646 en 1972. Concurrencé par l'EBCDIC d'IBM, il s'impose définitivement dans les années 1980.

Parfaitement adapté à l'anglais, il ne convient pas aux autres langues européennes, sans parler des langues sémitiques ou asiatiques.

Mais comme le travail ne se fait pas encore sur écran, on peut tricher en imprimant au même endroit un e puis un ' pour obtenir un é.

Un long processus de standardisation : ASCII

Dans les années 1980, le travail sur écran ne permet plus aux utilisateurs de l'alphabet latin étendu de composer aisément les caractères.

On adapte ISO-646 aux langues basées sur l'alphabet latin. Ce qui peut poser des soucis :

```
{ a[i]='\n'; }
```

pouvait devenir :

```
ä aÄiÜ='Ön'; ü
```

Un long processus de standardisation : ISO 8859

Au milieu des années 1980, le support des communications sur 8 bits est généralisé et 128 nouveaux caractères sont disponibles. Divers standards émergent sur chaque système. Dès 1986, l'ECMA publie un standard pour les langues européennes à écriture latine, grecque, cyrillique, arabe et hébraïque.
Après quelques évolutions, le standard est repris par l'ISO 8859 en 15 parties.
Pendant ce temps, Windows sort ses standards dérivés (notamment 1252), et Mac utilise des codages qui n'ont rien à voir.

Un long processus de standardisation : les langues asiatiques

Un alphabet japonais simplifié est standardisé en 1969 sur un octet (JIS X 201), puis étendu sur deux octets (Shift JIS). Le codage n'est pas complètement compatible avec ASCII : le \ est remplacé par le ¥, et le ~ est remplacé par le ¯.

128 caractères ne suffisent pas pour tous les idéogrammes chinois ! En 1980, le standard GB2312 code le chinois simplifié sur deux octets. Pour le chinois traditionnel de Hong-Kong et Taïwan, le Big5 est le standard de fait.

Mais des standards concurrents existent : EUC et ISO 2022.

Un long processus de strandardisation : Unicode

La multiplicité des codages nationaux compliquent les communications internationales. Le développement d'Internet pousse à la création d'un standard international unique et global : Unicode.

Objectif : un codage unique pour toutes les langues du monde. Jusqu'à 1 114 112 caractères possibles.

- 1991 : Unicode 1.0. 7 161 caractères et 24 "alphabets"
- janvier 2012 : Unicode 6.0. 110 181 caractères et 100 "alphabets"

Unicode : organisation

Les caractères sont codés sur 21 bits. Notation : U+xxxxxx.

Ils sont regroupés par blocs, qui sont regroupés en 17 plans de 16 bits. Le premier (U+0000 à U+FFFF) est le plan multilingue de base, qui permet d'écrire la plupart des langues actuelles.

Chaque caractère a un nom et éventuellement des caractères qui lui sont proches ou équivalents.
Exemples :

0043 C LETTRE MAJUSCULE LATINE C
→ 2102 ℂ c majuscule ajouré
→ 212D ℭ majuscule c gothique
→ 2201 ∁ complément

00E9 é LETTRE MINUSCULE LATINE E ACCENT AIGU
≡ 0065 e 0301 ◌́

Stockage d'Unicode

Il n'y a pas de manière naturelle de stocker 21 bits ! Pour les européens, la méthode la plus courante est UTF-8.

Tous les caractères d'Unicode sont codés en quelques octets :

de U+0000	à	U+007F	1 octet	ASCII
de U+0080	à	U+07FF	2 octets	latin, grec, cyrillique, hébreu, arabe
de U+0800	à	U+FFFF	3 octets	toutes les autres langues actuelles
de U+10000	à	U+1FFFFF	4 octets	les langues anciennes, les symboles

Glyphes

Unicode représente des alphabets complets. Il y a donc des caractères qui ont la même représentation graphique (*glyphe*).

Par exemple, les caractères :

- U+0041 (lettre majuscule latine A, bloc latin de base),
- U+0391 (lettre majuscule grecque Alpha, bloc grec et copte)
- U+0410 (lettre majuscule cyrillique A, bloc cyrillique)

sont tous représentés graphiquement A.

Même lettre, forme différente

En grec :

- la lettre sigma change de forme en fin de mot : μέσος
- la lettre beta change de forme en milieu de mot : βιϐῶν

En arabe, les lettres existent sous forme isolé, initiale, médiane et finale : ه هـه

Ligatures

Une ligature est un regrouprement esthétique de plusieurs lettres en un seul caractère.

On peut distinguer les ligatures linguistiques : *œ*, *æ*, *ij* (=*ij*), *ß* (=*ſs*) et les ligatures esthétiques : *ff*, *fi*, *fl*, *ffi*, *ffl*, *ſt*, *ct*.

Certaines ligatures deviennent des logogrammes : *&*, *@*, *⁊*, voire des lettres à part entières : *Ꝏ*.

En écriture devanāgarī : ॐ = अ+ऊ+ॅ

Ordre d'écriture

Dans les langues européennes, on écrit de gauche à droite : Bonjour !

Dans les langues sémitiques, on écrit de droite à gauche : שלום

En écriture devanāgarī, les lettres sont réordonnées dans certaines circonstances :
क+ ि+ क+ ी =किकी

Localisation ou Internationalisation ?

L'adaptation d'un logiciel à tous les contextes internationaux peut être plus ou moins poussée :

- *Localisation* : le logiciel est adapté pour une langue spécifique, mais ne permet pas d'en changer ensuite ;
- *Internationalisation* : le logiciel est adapté à un grand nombre de langues, mais n'en supporte qu'une seule à la fois ;
- *Multilinguisme* : le logiciel permet de manipuler plusieurs langues en même temps.

Locale

Le contexte linguistique est contenu dans les variables d'environnement suivantes :

- LANG : langue
- LC_NUMERIC : nombres
- LC_TIME : dates et heures
- LC_MONETARY : monnaie
- LC_PAPER : taille de papier
- LC_TELEPHONE : numéros de téléphone
- LC_MEASUREMENT : unités (système métrique ?)
- ...

Locale

Chacune de ces variables d'environnement prend une valeur de la forme : langue_PAYS.charset.
Exemples :

- en_US.UTF-8
- fr_FR.iso885915 = fr_FR@euro
- ru_RU.KOI8-R
- zh_CN.GB2312

gettext

gettext est une librairie utilisée pour internationaliser les codes. Son principe est de marquer dans le source les messages à traduire à l'aide de la fonction `gettext` ou son alias _.

L'utilitaire `xgettext` extrait les messages à traduire et crée un fichier `.pot` (Portable Object Template).

L'utilitaire `msginit` produit un fichier `.po` (Portable Object) par langue, qui sont complétés par les traducteurs.

L'utilitaire `msgfmt` compile les fichiers `.po` en fichiers `.mo` qui sont utilisés à l'exécution.

gettext par l'exemple

Le programmeur part d'un code à internationaliser :

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

print u'Hello, World.'
```

```
$ ./essai.py
Hello, World.
```

gettext par l'exemple

On commence par insérer les en-têtes gettext :

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

import gettext
gettext.bindtextdomain('essai', '.')
gettext.textdomain('essai')
_ = gettext.gettext

# TRADUCTEURS : Message de bienvenue
print _(u'Hello, World.')
```

gettext par l'exemple

On récupère les textes à traduire :

```
xgettext -L python -o essai.pot \
         --add-comments='TRADUCTEURS :' \
         essai.py
```

gettext par l'exemple

On obtient le fichier `essai.pot` suivant :

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2012-05-19 17:02+0200\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"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#. TRADUCTEURS : Message de bienvenue
#: essai.py:8
msgid "Hello, World."
msgstr ""
```

gettext par l'exemple

On remplit les informations manquantes et on transmet le fichier `essai.pot` au traducteur qui prépare un fichier `fr.po` :

```
msginit --locale=fr --input=essai.pot
```

gettext par l'exemple

Le traducteur remplit le fichier `fr.po` :

```
# Mon essai de gettext
# Copyright (C) 2012 Mathrice
# This file is distributed under the same license as the essai package.
# Stéphane Aicardi <Stephane.Aicardi@obspm.fr>, 2012.
#
msgid ""
msgstr ""
"Project-Id-Version: essai 1.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2012-05-19 17:14+0200\n"
"PO-Revision-Date: 2012-05-19 17:17+0200\n"
"Last-Translator: Stéphane Aicardi <Stephane.Aicardi@obspm.fr>\n"
"Language-Team: French\n"
"Language: fr\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=utf-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n > 1);\n"

#. TRADUCTEURS : Message de bienvenue
#: essai.py:8
msgid "Hello, World."
msgstr "Bonjour le monde !"
```

gettext par l'exemple

Il est alors compilé et installé :

```
mkdir -p fr/LC_MESSAGES
msgfmt -c -o fr/LC_MESSAGES/essai.mo fr.po
```

On teste :

```
$ ./essai.py
Bonjour le monde !
```

gettext par l'exemple

On peut accéder à plusieurs langues simultanément :

```
#!/usr/bin/env python
# -*- coding : UTF -8 -*-
import gettext
lang1 = gettext.translation('essai', '.', languages=['en'])
lang2 = gettext.translation('essai', '.', languages=['fr'])

# en anglais
lang1.install()
# TRADUCTEURS : Message de bienvenue
print _(u'Hello, World.')

# en français
lang2.install()
# TRADUCTEURS : Message de bienvenue
print _(u'Hello, World.')
```

On teste :

```
$ ./essai2.py
Hello, World.
Bonjour le monde !
```