# Unicode, tous les caractères

**Stéphane Bortzmeyer**

<bortzmeyer@nic.fr>

**Novembre 2003**

## Textes et informatique

Zone minée !

Les concepts sont nombreux, le vocabulaire complexe et peu intuitif.

`http://www.unicode.org/glossary/`

## Quelques rappels

- langue et écriture sont des choses différentes (le turc est passé de l'alphabet arabe au latin au début du siècle dernier)
- le concept de caractère n'est pas trivial (ligatures, diacritiques, etc)
- les autres ne font pas comme nous (écriture en lacet, absence de casse, sans compter les idéogrammes)

## Généralités

Pour gérer des textes dans un programme, il faut :

- Un répertoire (une liste, une table de caractères)
- Un encodage (une représentation en mémoire. L'IETF dit *on the wire*)

Pendant longtemps, les normes mélangeaient les deux.

## Répertoire et encodage

Unicode les sépare.

On ne peut pas dire “Unicode est un jeu de caractères 16 bits”.

## Et la forme ?

Unicode traite des caractères abstraits, pas des formes (glyphes).

Unicode n'est pas une police de caractères.

## Pourquoi Unicode ?

Il existe déjà beaucoup de jeux de caractères, souvent largement implémentés.

Unicode est le seul qui permette des textes multi-écritures.

Unicode est le seul qui permette de traiter toutes les écritures.

## US-ASCII

Alphabet latin, sans décorations rigolotes.

Répertoire : de 0 à 127.

Encodage : un caractère par unité de stockage (un octet, typiquement).

Ne convient qu'aux anglophones et aux protocoles IETF (HELO, VRFY, MAIL FROM).

## ASCII accentué

(Historique, pour ceux qui se souviennent des imprimantes 7 bits)

Répertoire de 0 à 127 mais certains caractères comme { ou } remplacés par des caractères composés.

Pratique pour le français, gênant pour le C ou le Perl.

## ISO-8859-X

Un ensemble de normes ISO utilisant des encodages sur 8 bits.

Exemple : ISO-8859-1 (dit *Latin-1*). `man iso_8859_1`

Répertoire : de 0 à 255

Encodage : un caractère par unité de stockage (un octet).

## Écriture grecque

ISO-8859-7

La partie haute utilise l'alphabet grec, la partie basse l'US-ASCII (pratique pour le NIC grec).

Et si on veut écrire un texte mixte, français et grec ?

Et si je reçois un document, comment savoir s'il est ISO-8859-1 ou bien ISO-8859-2 ?

## Autres jeux de caractères

KOI-8 (cyrillique), Big5 (chinois), MacOS (différent du Latin-1).

Babel

## Unicode fait comment ?

Unicode définit un répertoire, une liste.

(5) Jeu de caractères abstrait

## Unicode est juste une table

| Nom | Glyphe |
|---|---|
| Latin capital A | A |
| Latin capital C cedilla | Ç |
| Greek small Alpha | $\alpha$ |
| Hebrew Alef | ℵ |
| Arabic Alef | ? |
| Oriya DDHA | ? |

## Unicode numérote

### (4) Jeu de caractères codés

Chaque “caractère” a un index, le point de code (*code point*), noté en hexa, et un nom.

## Unicode est juste une table

| *code point* | Nom | Glyphe |
|---|---|---|
| U+0041 | Latin capital A | A |
| U+00C7 | Latin capital C cedilla | Ç |
| U+03BB | Greek small Alpha | $\alpha$ |
| U+0500 | Hebrew Alef | ℵ |
| U+0627 | Arabic Alef | ? |
| U+0B22 | Oriya DDHA | ? |

## Compatibilité

Coïncidence : les 128 premiers index sont identiques à ceux d'US-ASCII. Les 128 suivants à Latin-1.

## L'index n'est pas la représentation

La représentation dépend de l'encodage.

## Encodages pour Unicode

### (3) Forme codée en mémoire

Unicode permet plusieurs encodages. Les plus connus :

- UTF-8 (RFC 2279), US-ASCII en est un sous-ensemble. Encodage par défaut en XML.
- UTF-16,
- UTF-32, le seul où index = représentation mais le plus consommateur de mémoire

## UTF-8, l'encodage de l'Internet ?

Le plus répandu. Plein de propriétés intéressantes (comme la limitation de la contextualité).

Exemple : l'octet 0x61 (petit a) n'apparaitra **que** dans un petit a.

```
Code point (hex.)   UTF-8 octet sequence (binary)
0000 0000-0000 007F 0xxxxxxx           US-ASCII
0000 0080-0000 07FF 110xxxxx 10xxxxxx
0000 0800-0000 FFFF 1110xxxx 10xxxxxx 10xxxxxx
```

## UTF-8 a aussi des défauts

Par contre, récupérer le Nème caractère est inefficace (le nombre de *code units* par point de code est variable).

## Trouver l'encodage

Sur le réseau, il faut "taguer" (en-tête MIME).

En local, un système de stockage donné (SGBD, système de fichiers) doit avoir un encodage standard.

## Les couches basses

### (2) Mécanisme de sérialisation des caractères

Petit-boutien ou gros-boutien ?

### (1) Surcodage de transfert

Compression et/ou Base 64 par dessus.

## Les diacritiques

Un point de code peut se combiner avec un autre (diacritiques).

Le *code point* n'est **pas** ce que l'utilisateur appelerait un caractère. é = e + ’

Il y a aussi les précomposées les plus courantes comme ç.

## Supporter Unicode

Qu'est-ce que cela implique exactement ?

Au choix, en partant du plus facile :

- Stocker et restituer l'Unicode,
- E/S avec d'autres encodages (codecs),
- Tri, *case folding*, recherche de regexp, etc corrects,
- Afficher (nécessite une grosse fonte !), le moteur de rendu est forcément complexe.

## Casse

Unicode fournit des tables de *case folding*.

Pas évident car un caractère peut avoir une chaine comme majuscule (ß donne SS).

Peut aussi dépendre de la *locale*.

## Tri

Même genre de problèmes que la capitalisation.

ä vient après a (France) ou bien après z (Suède) ?

## Normalisation

Signatures cryptographiques, unicité (noms de domaines, fichiers) et d'autres nécessitent une normalisation.

L'excellent programme `http://www.w3.org/International/charlint/`

## Plusieurs normalisations

1. NFC (combine au maximum)
2. NFD (décombine au maximum)
3. etc

On peut aussi définir des normalisations au dessus de celles-ci (“stringprep” de l’IETF, RFC 3454).

## charlint

Avant normalisation :

```
% produce-data.pl | consume-data.pl
LATIN SMALL LETTER C (Basic Latin)
COMBINING CEDILLA (Combining Diacritical Marks)
LATIN SMALL LETTER E (Basic Latin)
COMBINING ACUTE ACCENT (Combining Diacritical Marks)
```

Après normalisation NFC :

```
% produce-data.pl | charlint.pl | consume-data.pl
LATIN SMALL LETTER C WITH CEDILLA (Latin-1 Supplement)
LATIN SMALL LETTER E WITH ACUTE (Latin-1 Supplement)
```

## Normalisation et politique

Les chaines suivantes sont-elle équivalentes :

- cafe
- café
- câfé
- cafe’
- CAFE
- CAFÉ

?

## Compter les caractères

Que doit renvoyer
`strlen(ma_chaine_unicode)` ?

- Le nombre de graphèmes (“caractères” pour l'utilisateur) ?
- Le nombre de *code points* (“caractères” pour l'informaticien) ?
- Le nombre de *code units* (octets) ?

## Démonstration de la différence

"character semantics" vs. "byte semantics" en Perl

```
print $ustring->length(), "\n"; # Affiche le
                  # nombre de code points

$utf8 = $ustring->utf8;
print length($utf8), "\n"; # Affiche le nombre
                  # d'octets
```

## Sécurité

### Valider les entrées

Valider positivement (caractères autorisés) et pas négativement (caractères interdits)

Valider **après** normalisation.

Attention en écrivant un décodeur (UTF-8, par exemple)

## Depuis un programme

Presque tous les langages de programmation savent faire de l'Unicode...

... plus ou moins bien.

L'encodage interne peut être UTF-8 (Perl), UTF-16 (Java) ou sélectionnable à la compilation (Python). Attention aux E/S binaires !

## Stocker de l'Unicode

Support Unicode dans les SGBD : ce n'est pas tout de stocker et de restituer. Et le ORDER BY ? Et les E/S ?

XML est Unicode dès le début : tout analyseur XML doit accepter UTF-8 et UTF-16.

## Dans les normes

Comme XML, toutes les normes devraient prévoir Unicode dans les chaines de caractère.

## Exemple : IDN

Internationalized Domain Names

`ping www.académie-française.fr`

RFC 3490 à 3492.

Le nom Unicode est normalisé (“nameprep”).

Puis encodé en “punycode”
(`www.xn--acadmie-franaise-npb1a.fr`).

## IDN pratique

Déjà dans certains TLD.

Plein de logiciels libres, mutt, Mozilla, bibliothèques comme GNU libidn.

## Outils

- charlint pour normaliser
- recode pour convertir en batch
- emacs, vim ou yudit pour éditer

## Organisations

– Consortium Unicode
  `http://www.unicode.org/`
– Groupe ISO qui sort 10646.

Les deux normes sont compatibles, les parasites de l'ISO ne font que mettre un tampon.

Le Consortium Unicode traite en plus les problèmes concrets (comme les règles de changement de casse).

## Malheureusement, les licences

Licence de la norme toujours très restrictive
(mais moins qu'à l'ISO).

## Le déploiement

Unicode, c'est comme IPv6

- Problème d'oeuf et poule
- Les plus puissants sont les moins intéressés
- Le poids de l'existant (fichiers...)