Encodage de texte

BRUNO DARID

3 décembre 2019

PLAN

1 L'objectif 1

2 L'ASCII 1

3 Évolution : la norme ISO 8859 1

4 Norme ISO 10646 - Standard Unicode 2

5 Principe de l'encodage UTF-8 2

6 Unicode et python 2

1 L'objectif

Il s'agit d'associer un caractère à un numéro unique : c'est ce que l'on appelle un **encodage**. Pour être efficace un encodage doit représenter le plus de caractères possible et être compact.

2 L'ASCII

À l'origine plusieurs encodages incompatibles entre eux coexistaient jusqu'à ce qu'une norme s'impose dans les années 60 : L'**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Un octet est utilisé pour coder un caractère. En réalité seuls 7 bits sont utilisés : on pouvait donc coder $2^7 = 128$ caractères avec cette norme. Le dernier bit était utilisé à des fins de contrôle lors des transmissions.
Il faut noter que seuls les caractères « anglo-saxons »étaient présents. Les caractères accentués n'étaient pas représentables.

3 Évolution : la norme ISO 8859

Afin d'intégrer plus de caractères (*notamment les caractères accentués*), une première évolution a eu lieu : la norme ISO 8859. Il s'agit d'une extension de l'ASCII : on utilise le 8ème bit (utilisé jusque là pour le contrôle de parité) pour coder des caractères. On a ainsi $2^8 = 256$ possibilités ! Une table de correspondance code hexadécimal ou décimal $\leftrightarrow$ ASCII et ASCII étendu peut être trouvée à cette adresse

4 Norme ISO 10646 - Standard Unicode

Bien qu'intéressante l'évolution vers la norme ISO 8859 ne règle pas tous les problèmes et notamment des langues autres qu'européennes (*asiatiques entre autres*). Pour tenir compte de tous les caractères utilisés, l'ISO proposa une nouvelle norme. Il s'agit alors de définir un caractère comme une entité possédant :

— un nom ;

— un numéro, appelé encore **point de code** et noté U+xxxx où xxxx est un nombre hexadécimal.

Un consortium privé à but non lucratif, **Unicode**, proposa alors plusieurs techniques pour encoder ces points de code. La plus utilisée est manifestement l'encodage UTF-8 (**U**niversal **T**ransformation **F**ormat) ; le « 8 »signifie qu'il faut au minimum 8 bits pour représenter un point de code.
Les encodages UTF-16 (complexe) et UTF-32 (peu économe) ne seront pas abordés ici.

5 Principe de l'encodage UTF-8

Cet encodage permet de coder un caractère sur 1, 2, 3 ou 4 octets. Si le point de code est inférieur ou égal à 127 (7F en hexadécimal) le caractère est codé sous la forme $0xxxxxxx$ comme en ASCII.
Sinon, on utilise la forme binaire du point de code et on détermine ainsi le nombre de bits nécessaire au codage du caractère.

Point de code	Octets (binaire)	Nombre de bits nécessaire
U+0000 à U+007F	0xxx xxxx	7
U+0080 à U+07FF	110x xxxx 10xx xxxx	11
U+0800 à U+FFFF	1110 xxxx 10xx xxxx 10xx xxxx	16
U+10000 à U+10FFFF	1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx	21

L'octet de poids fort est composé d'une séquence de 1 terminé par un 0. Le nombre de 1 présent dans cette séquence indique le nombre d'octet utilisé pour coder le caractère.

6 Unicode et python

Les chaines de caractères en python utilisent l'encodage UTF-8. D'ailleurs, on peut saisir directement une chaine à partir de ses points de code (si on les connait !), en prenant la précaution de les préfixer de `\u`.

```
In [2]: ch = '\u0045\u0020\u2260\u0020\u2205'
        print(ch)
```

```
E ≠ ∅
```

Il est possible en python de connaître le résultat de l'encodage (*en hexadécimal*), en utilisant la méthode `encode`.

```
In [3]: chaine_octets = ch.encode('utf8')
        print(chaine_octets)

b'E \xe2\x89\xa0 \xe2\x88\x85'
```

Remarques

— On voit apparaître un nouveau type disponible en python : le type **bytes** (*octet*) ;

— La chaine d'octets et la chaine de caractères n'ont pas même longueur : en UTF-8, un caractère peut être codé sur plusieurs octets.

On peut également faire le conversion inverse (*d'une chaine d'octets vers une chaine caractères*) avec la méthode `decode`.

```
In [4]: b'E \xe2\x89\xa0 \xe2\x88\x85'.decode('utf8')

Out[4]: 'E ≠ ∅'
```