# Table des matières

~~stoggle_buttons~~

# Principes de base

- Stockage en binaire
- L'unité de stockage est l'octet
- Ecriture des valeurs en hexadécimal
- si stockage d'une donnée sur plusieurs octets : "endianness". Il faut préciser ce qui est stocké à l'adresse la plus basse en mémoire : l'octet de poids fort ou l'octet de poids faible ? Par extension, cela caractérise l'ordre dans lequel sont stockés les octets d'une donnée dans un fichier. C'est à cette occaion, qu'on rencontrer le terme BOM. Infos intéressantes ici : http://fr.wikipedia.org/wiki/Endianness

# Chaînes de caractères

## Unicode

### Le site Officel : très fourni

- Le site Unicode
- La dernière version du standard
- L'algorithme d'Unicode. On y comprend le mode de fonctionnement et les problèmes qui se sont posés aux concepteurs du système (représentation, tris, ...)

### Un site plus simple et plus utilitaire

- Les "sections" de code Unicode
- Des exemples de jeux de caractères répertoriés par Unicode
- Des outils d'encodage-décodage
- La table des caractères de 0000 à 007F

### UTF-8

Un exemple :

Vous voulez représenter, en UTF-8, le symbole "Angstrom". Il faut partir de son code Unicode.

- Vous cherchez sur le site Unicode et vous trouvez :

| | | |
|---|---|---|
| Å | U+212B | ANGSTROM SIGN |

```
Son code Unicode est : 212B.
```

- Cette valeur 212B est comprise entre 0800 et FFFF (voir le tableau ci-dessous).

```
Donc la réprésentation se fera sur 16 bits qui seront stockés sur 3 octets
(voir le tableau ci-dessous)
au format indiqué dans les 6 colonnes de droite du tableau.
```

- Vous traduisez le code hexadécimal en binaire :

```
212B = 0010 0001 0010 1011
```

Ensuite, vous aurez besoin du tableau de référence de la tranformation (voir en bas de page). Dans la suite du document, en **gras** : les bits ajoutés par la transformation UTF-8, les autres bits sont issus du code Unicode.

- Il faut décomposer les 16 bits du code Unicode en 3 blocs de 4, 6 et 6 bits : 0010, 000100 et 101011
- Puis ajouter les en-têtes des 3 octets : **1110**, **10** et **10**. Pour obtenir : **1110**0010 , **10**000100 , **10**101011

```
 Le nombre de bits de l'en-tête du premier octet indique le nombre d'octets
de la transformation.
```

- Le code de la transformation UTF-8 sera : E2 84 AB
- Une page où vous trouverez un convertisseur "tout fait" : http://www.ltg.ed.ac.uk/~richard/utf-8.cgi?input=212B+&mode=hex

## La table de correspondance

| **Bits of** | **First** | **Last** | **Bytes in** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **code point** | **code point** | **code point** | **sequence** | **Byte 1** | **Byte 2** | **Byte 3** | **Byte 4** | **Byte 5** | **Byte 6** |
| 7 | U+0000 | U+007F | **1** | 0xxxxxxx | | | | | |
| 11 | U+0080 | U+07FF | **2** | **11**0xxxxx | **10**xxxxxx | | | | |
| 16 | U+0800 | U+FFFF | **3** | **111**0xxxx | **10**xxxxxx | **10**xxxxxx | | | |
| 21 | U+10000 | U+1FFFFF | **4** | **1111**0xxx | **10**xxxxxx | **10**xxxxxx | **10**xxxxxx | | |
| 26 | U+200000 | U+3FFFFFF | **5** | **11111**0xx | **10**xxxxxx | **10**xxxxxx | 10xxxxxx | 10xxxxxx | |
| 31 | U+4000000 | U+7FFFFFFF | **6** | **111111**0x | **10**xxxxxx | 10xxxxxx | 10xxxxxx | 10xxxxxx | 10xxxxxx |