Jeux de caractères

Jean-Marc Bourguet

1 Introduction

Ce document, pour le moment encore incomplet, a pour vocation de rassembler mes connaissances en ce qui concerne les jeux de caractères codés.

2 Définitions

caractère (*character)*) c'est une unité de représentation textuelle de donnée. La lettre A par exemple, ou le chiffre 5 ou le symbole +.

glyphe (*glyph*) c'est la représentation dessinée du caractère. Un ensemble de glyphes, c'est une *fonte* (*font*).

code (*code*), c'est un nombre représentant un caractère.

jeux de caractères codés (*coded character set, codeset, charset, codepage*) c'est une table de correspondance entre des codes et des caractères.

point de codage (*codepoint*) c'est un nombre faisant partie de l'ensemble dans lequel les codes des caractères d'un codeset sont choisis. Tous les points de codage ne correspondent pas nécessairement à des caractères.

encodage (*encoding*) c'est une manière de décoder une suite de bytes en une séquence de caractères (ou inversément d'encoder une séquence de caractères en une suite de bytes).

multiplet (*byte*) c'est l'unité élémentaire de stockage utilisée par un encodage.

3 Caractère et glyphe

La relation entre caractère et glyphe n'est pas toujours aussi simple que la définition peut le laisse croire.

On peut avoir des glyphes qui représentent plusieurs caractères (« fi » dans la fonte utilisée pour ce document est un seul glyphe représentant vraissemblablement deux caractères ; et « A » pourrait être le glyphe à la fois pour la lettre latine A majuscule et pour la lettre grecque alpha majuscule).

On peut avoir plusieurs glyphes pour un même caractère. Dans des fontes différentes (« A » et « $\mathscr{A}$ » par exemple) ou dans une même fonte (σ et ς sont des glyphes présentes dans la même fonte représentant le même caractère : la lettre grecque sigma).

Le cas du sigma est intéressant, on peut aussi facilement imaginer un charset où les deux variantes du sigma sont présentes comme un charset où un seul sigma est présent, laissant le choix à un moteur d'affichage connaissant les règles du grec (l'une forme se trouve à la fin des mots, l'autre au début ou à l'intérieur).

Les lettres accentuées offrent le même genre de choix. Certains charsets les forment par composition (Unicode par exemple a des « caractères combinants » qui modifient le caractère qui suit et qui sont utilisés pour les accents, d'autres charsets utilisent des séquences « accent, backspace lettre ») d'autres ont les lettres accentuées comme caractères.

Donc en dernier ressort, c'est le charset utilisé qui va définir ce qu'est précisément un caractère.

4 Caractère et point de codage

Tous les points de codage ne correspondent pas nécessairement à des caractères. Certains peuvent être laissés libres, soit parce que l'ensemble choisi pour les points de codage est très vaste (c'est le cas de celui d'Unicode par exemple), soit pour être compatible avec d'autres spécifications (par exemple Unicode ne définit presqu'aucun point avec les codes de 0 à 31, les laissant libre pour les fonctions définies par ECMA-48).

D'autres point de codage peuvent être utilisés pour des usages autres que la représentation de données. Par exemple pour commander des périphériques. On parle parfois alors de *caractères de contrôle*. Certains caractères de contrôle sont aussi parfois utilisés pour structurer le texte (par exemple Unix utilise le line feed comme marque de fin de ligne).

5 Code et encodage

L'encodage le plus simple représente simplement chaque caractère par sa valeur. Un encodage un peu plus compliqué, c'est d'avoir deux charsets disjoints. C'est le cas des encodages les plus utilisés pour les charsets 8 bits qui ne spécifient généralement rien au sujet des caractères de contrôle.

Mais on peut avoir des encodages plus sophistiqués. Unicode est un exemple qui prévoit de base cinq encodages offrant des rapports densité/facilité d'utilisation différents. Un autre exemple est MIME qui prévoit plusieurs encodages différents sur 7 bits pour les charsets sur 8 bits.

Un dernier exemple est la norme ECMA-43, qui prévoit des séquences pour passer d'un charset à un autre.

6 Charsets

L'utilisation de jeux de caractères codés commence avec le telex (le code morse utilisé par le télégraphe ne correspond pas à la définition donnée) et continue avec les codes Holerith utilisés dans les tabulatrices, ces systèmes généralement électromagnétiques utilisant les cartes perforées pour faire les traitements maintenant faits par les ordinateurs.

Les premiers ordinateurs utilisaient soit des charsets compatibles avec ceux des tabulatrices vendues par leur fabriquant (c'est le cas du BCD pour IBM), soit des charsets particuliers à la machine. La nécessité d'une normalisation s'est vite faite sentir.

Les contraintes de l'époque rendaient pratiquement impossible la solution d'un jeu de caractères universel vers lequel on tend pour le moment avec Unicode. Pourtant, il était déjà souhaitable d'avoir des points communs entre les charsets utilisés. On s'est donc retrouvé avec trois types de solutions cohabitant :

- une indication externe (on dit « hors bande » dans le jargon) indiquant quel charset utiliser ;
- l'utilisation de charsets ayant un ensemble de caractères en commun et codés de la même façon ;
- l'utilisation d'encodage permettant de changer de charset.

6.1 Charsets sur 7 bits

6.1.1 ASCII

C'est un charset sur 7 bits défini aux États-Unis en plusieurs étapes durant les années 60. Ce charset a servi de base à la famille ISO-646 et la dernière étape a été la modification du charset pour l'adapter comme variante nationale US de ISO-646.

6.1.2 ISO-646

C'est une famille de charsets basée sur l'ASCII dans lequel remplace certains caractères. Le charset ASCII a d'ailleurs été redéfini par après dans le cadre de cette norme comme étant la variante nationale US.

6.2 Charsets sur 8 bits

6.2.1 EBCDIC

C'est une famille de charset sur 8 bits définie par IBM a peu près au moment où la normalisation de l'ASCII commencait. Elle a été conçue pour être très proche du charset le plus utilisé alors sur des machines IBM : BCD. Cela a eu quelques conséquences dont la plus génante vraissemblablement est que les lettres n'ont pas leurs points de codage consécutifs.

6.2.2 ISO-8859

C'est une famille de charsets basée sur l'ASCII. Tout le charset ASCII est disponible avec le 8^{e}bit à 0. Les 128 caractères restant sont définis comme étant 32 caractères de contrôle supplémentaires. Le reste dépendant de la variante utilisée.

6.3 ISO-10646 et Unicode

Avec l'augmentation de la mémoire disponible et l'internationalisation croissante, il est devenu de plus en plus pertinent de chercher à avoir un charset universel, contenant tous les caractères. C'est un jeu de caractère codé sur un peu moins de 21 bits.

ISO-10646 est une norme internationale cherchant à construire ce répertoire universel. Unicode est un standard [1] défini par un consortium avec le même objectif. Heureusement, les deux sont coordonnés et utilisent les mêmes points de codage pour les mêmes caractères. La différence majeure est qu'Unicode a des objectifs plus étendus qu'ISO-10646 et défini plus précisément le comportement des applications qui veulent être conforme à ce standard.

Ce charset est compatible avec ASCII (les 128 premiers points de codage codent les mêmes caractères) et avec ISO-8859-1 (les 128 points de codage suivant code les mêmes caractères).

7 C et C++

Passons maintenant au C et au C++. Le type char est utilisé pour représenter des bytes au sens ci-dessus. Ces langages imposent en plus comme contrainte qu'un byte fasse au moins 8 bits et qu'il soit la plus petite unité adressable. Si on a une machine adressable par mot (ça devient très rare), on peut soit prendre pour byte un mot complet, soit utiliser une sous-division du mot avec des pointeurs qui contiennent une information sur la sous-division à utiliser. Si on a une machine adressable par bit (je n'en connais qu'une qui ait eu une vocation d'usage générale), on ne pourra pas profiter de la chose.

Les chiffres de 0 à 9 doivent avoir des points de codage consécutif (donc '9'-'0' vaut 9). Les lettres peuvent ne pas avoir des points de codage consécutif (donc on peut utiliser de l'EBCDIC).

Tous les caractères du jeu de base doivent avoir des points de codage positifs (donc si on utilise de l'EBCDIC, char doit être non signé).

Le charset et l'encodage à utiliser pour interpréter la valeur d'un byte comme caractère fait partie de la « locale ». Je ne rentrerai pas pour le moment dans les détails sur les locales. Il y a quelques contraintes : il faut que tous les caractères du jeu de base (en gros, ceux qui servent dans la syntaxe) soit représentable par un seul byte et le même dans tous les encodages supportés et que dans tous les encodages ce byte représente toujours le caractère du jeu de base quel que soit le contexte.

1. J'utilise la nuance permise par le français entre *norme* définie par un organisme officiel et *standard* défini par une organisation moins officielle ou décrivant simplement un état de fait

Le type wchar_t est utilisé pour représenter des codes. Il faut qu'il soit assez grand pour contenir tous les codes du charset en ayant le plus. Il faut à nouveau que les caractères du jeu de base soient représentés par le même code dans tous les charsets.