Date : 03/11/2006

L'application d'Unicode à UNIMARC : espoirs et réalités

Olga Zhlobinskaya
National Library of Russia

Traduction : Denis Jouniaux
(Bibliothèques communales de Schaerbeek, Belgique)
DJouniaux@schaerbeek.irisnet.be

Meeting:	**77 UNIMARC**
Simultaneous Interpretation:	**Yes**

WORLD LIBRARY AND INFORMATION CONGRESS: 72ND IFLA GENERAL CONFERENCE AND COUNCIL
20-24 August 2006, Seoul, Korea
http://www.ifla.org/IV/ifla72/index.htm

L'utilisation d'Unicode pour coder les caractères de pratiquement tous les langages écrits connus semble être un instrument de premier ordre pour les systèmes utilisés en bibliothèque puisqu'il permet (en tout cas nous aimons le croire) un support multi écriture pour les enregistrements MARC. En fait, Unicode offre plutôt un *potentiel* pour support multi écriture plutôt qu'une solution toute faite, et pour utiliser ce potentiel, nous devons résoudre une série de nouveaux problèmes causés par l'arrivée d'Unicode.

En parlant d'Unicode, nous devons nous rappeler qu'Unicode est avant tout une table de codes allouant des nombres aux caractères. Pour que les ordinateurs puissent comprendre les séquences de ces nombres, chaque caractère devrait être représenté en une seule séquence d'octets. Unicode a défini un certain nombre de formats d'encodage de caractères, les plus largement utilisés étant UTF-8, UTF-16 et UTF-32.

UTF-8 est le format de transformation Unicode qui permet de représenter les valeurs UNICODE en séquences de longueur variable d'octets de 8 bits. UTF-8 transforme chaque caractère Unicode en une séquence d'un à quatre octets. Les caractères ASCII sont codés en un seul octet. Tous les autres caractères sont codés à l'aide de 2 octets ou plus. Les bits les plus signifiants du premier octet d'une séquence multi octets déterminent combien d'octets suivent pour ce caractère. En théorie, 2 exposant 31 caractères peuvent être encodés, mais en pratique, environ 96 000 positions de code sont utilisées. Ainsi, UTF-8 est compatible avec ASCII – tout fichier qui contient uniquement des caractères ASCII aura le même codage sous ASCII et UTF-8, préservant ainsi les éléments structurels de base de l'enregistrement MARC. D'un autre côté, UTF-8 est capable de représenter n'importe quel caractère Unicode.

En plus de sa compatibilité descendante avec l'ASCII, il y a une autre caractéristique d'UTF-8 méritant d'être mentionnée. Pour tout texte contenant principalement des caractères ASCII, UTF-8 permet d'économiser de l'espace de stockage, ce qui peut être d'une certaine importance en ISO 2709.

Si on utilise UTF-16 et UTF-32, chaque caractère est codé avec des éléments de combinaison de code de respectivement 2 ou quatre octets. Toute chaîne multi octet peut être mémorisée avec l'octet les plus signifiant placé en premier ou en dernier. Dans le premier cas, il est appelé gros-boutiste, dans l'autre petit-boutiste. Afin de permettre au système de réception de détecter l'ordre des octets dans la chaîne reçue, chaque entité Unicode devrait commencer par le caractère connu comme « balise d'ordre d'octets » (BOM), servant à indiquer que c'est un fichier Unicode, et le format utilisé. Grâce au BOM, le système peut toujours distinguer si ce format est UTF-16BE, UTF-16LE, UTF-32BE ou UTF-32LE. Tout comme pour UTF-8, c'est toujours le même ordre d'octets, et si le BOM initial est utilisé, c'est *seulement* une indication que le fichier texte est en UTF-8.

Les enregistrements MARC multi écriture ont été pendant longtemps un rêve inaccessible pour la communauté des bibliothécaires, et le désir d'utiliser toutes les opportunités offertes par l'apparition d'Unicode est bien compréhensible. De nos jours, de plus en plus de vendeurs déclarent que leurs systèmes offrent un support totalement compatible avec Unicode. Mais quelles sont les perspectives offertes par une telle intervention de grande envergure d'Unicode ?

Comme nous l'avons déjà dit, UTF-8 est une sorte de pont d'ASCII vers Unicode. Il nous permet d'indiquer dans un enregistrement standard quel jeu de caractères est utilisé et le système peut reconnaître cette indication correctement et traiter les données dans l'enregistrement correspondant. Nous pouvons donner une telle indication avec un mécanisme UNIMARC usuel – en indiquant code 50 dans le champ 100$a, pos. 26-27, ou, si c'est en MARC21, avec un code dans l'amorce de bande (pos. 9).

Le problème apparaît avec la transition vers UTF-16 et UTF-32. Mais avant de se précipiter pour trouver une solution à un problème, il est d'usage de réfléchir. Qu'en est-il exactement ?

En UTF-16 et UTF-32, un jeu de caractères indicatifs dans le champ 100 n'aurait pas d'effet, étant donné que, lisant le champ et l'enregistrement comme un ensemble, le système doit reconnaître dans quel environnement il travaille, que ce soit UTF-8, UTF-16 ou UTF-32. Dès lors, si le système le sait au préalable, à quoi bon indiquer un jeu de caractères dans l'enregistrement ?

Examinons quelques solutions possibles.

(a) la solution retenue en MARC21, c'est-à-dire l'indication de l'Unicode utilisé dans l'amorce de bande, ne peut pas être considérée comme absolue – oui, l'information sur le jeu de caractères utilisé est maintenant contenue dans l'amorce de bande plutôt que dans le corps de l'enregistrement, mais pour lire et interpréter l'amorce de bande, le système doit toujours connaître le jeu de caractères – donc, nous en revenons à notre point de départ.
(b) Opérer des changements sur ISO 2709, qui est encore aujourd'hui le contenant les plus utilisé pour les enregistrements MARC.

ISO 2709 fut développé quand tout caractère pouvait être représenté avec un et un seul octet. Depuis, il a été considéré nécessaire de réviser le standard pour incorporer des spécifications pour l'utiliser pour des enregistrements avec Unicode. Selon l'avant-projet de révision d'ISO 2709, le terme « octet » est défini ; l'étiquette d'enregistrement est fixée en longueur à 24 octets, chacun représentant un caractère, quelque soit le nombre d'octets utilisés pour un caractère dans l'enregistrement ; finalement, il est déterminé que le codage de base des caractères est ISO 646 ou ISO 10646 avec codage UTF-8, c'est-à-dire que l'enregistrement annonce et de répertoire, les indicateurs, les identifiants sous-zone, caractères de fin de champ et de fin d'enregistrement utilisent un octet par codage de caractère. Donc, le nouvel ISO est supposé clarifier l'utilisation d'Unicode avec le codage UTF-8. Il ne dit rien à propos d'UTF-16 ou UTF-32, et cela semble absolument correct, depuis qu'ISO 2709 décrit la structure de l'enregistrement, et le codage n'est pas une question de syntaxe. Maintenant posons la question suivante : pouvons-nous utiliser deux UTFs différents dans le même enregistrement, c'est-à-dire UTF-8 pour les éléments structurels principaux, comme prescrit par ISO 2709, et UTF-16 (ou UTF-32) pour les zones de données ? Cela semble étrange pour l'auteur ; en tout cas, ce point devrait être formulé de façon explicite dans le format afin d'éviter les malentendus.

Mais ISO 2709 n'est pas le seul contenant pour la transmission de données bibliothéconomiques, comme c'était le cas auparavant. XML, en particulier, peut être utilisé à cette fin, et nombre de schémas ont déjà été développés (MARCXML, UNIMARC XML Slim). Il est évident que la solution ci-dessus évoquée ne fonctionnerait pas pour XML, et dans ce cas, nous avons besoin d'une autre solution.

Bien que les codages par défaut des jeu de caractères XML soient UTF-8 et UTF-16, des codages spécifiques pour des documents XML peuvent être définis dans la déclaration initiale XML pour tout le document ou entité (qui peut être considérée comme une partie mémorisée séparément de l'ensemble du document). En complément du BOM, indiquant exactement quel codage est utilisé dans l'enregistrement, cette déclaration semble être suffisante pour que le système récepteur interprète l'information.

On pourrait dire –OK, s'il n'y a pas de problème avec XML, oublions ISO 2709 et utilisons XML. Mais nous devrions nous rappeler que la plupart des utilisateurs des enregistrements MARC utilisent toujours ISO 2709 plutôt qu'XML, et nous ne pouvons pas les laisser tomber.

Maintenant, il est temps de nous arrêter un moment pour décider si nous allons nous en sortir avec une série de solutions propres à chaque type de contenant, ou chercher une solution qui serait applicable à la fois à ISO 2709, XML et, si possible, toute autre syntaxe.

Nous sommes convaincus que la solution devrait être générale, bien que pour l'instant nous ne pouvons pas proposer un instrument non spécifique à un contenant prêt à l'emploi, et nous ne savons pas si quelqu'un peut le faire dès maintenant. Mais y a-t-il réellement urgence à trouver une solution concrète immédiatement ? Dans une telle situation, nous ne pouvons pas garantir que la meilleure décision serait prise. Pour autant, en pratique, nous ne sommes pas encore tombés sur une situation où il est nécessaire d'utiliser des codes caractères qui ne pourraient pas être représentés avec UTF-8, ou bien que l'imprimé de codage ne fonctionne pas pour toute autre raison. Nous pensons que cette réalité nous donne du temps afin de ne pas prendre une décision trop rapide.