Groupe de travail Réseau
Request for Comments : 5051
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

M. Crispin, Université de Washington

octobre 2007

i;unicode-casemap - algorithme simple d'interclassement pour chaînes Unicode

Résumé
Le présent document décrit "i;unicode-casemap", un simple interclassement insensible à la casse pour les chaînes Unicode. Il assure des opérations d'égalité, de sous chaîne, et d'ordre.

Table des Matières

1. Introduction

La collation "i;ascii-casemap" décrite dans la [RFC4790] est assez simple à mettre en œuvre et fournit des comparaisons indépendantes de la casse pour les 26 lettres de l'alphabet latin. Elle est spécifiée comme le comparateur par défaut et/ou de base dans certains protocoles d'application, par exemple, de la [RFC5256].

Cependant, la collation "i;ascii-casemap" ne produit pas de résultats satisfaisants avec les caractères non ASCII. Il est possible, avec une extension modeste, de fournir une collation plus sophistiquée avec une plus grande applicabilité multilingue que avec "i;ascii-casemap". Cette extension fournit des comparaisons indépendantes de la casse pour un bien plus grand nombre de caractères. Elle interclasse aussi les caractères à signes diacritiques avec les formes de caractères sans diacritiques.

Cette collation, "i;unicode-casemap", est destinée à être une solution de remplacement préférée à i;ascii-casemap". Elle ne remplace pas la collation "i;basic" décrite dans [BASIC].

2. Description de collation Casemap Unicode

La collation "i;unicode-casemap" est une simple collation qui est insensible à la casse dans son traitement de caractères. Elle assure des opérations d'égalité, de sous chaîne, et d'ordre. L'opération de vérification d'égalité retourne "valide" pour toute entrée.

Cette collation permet des chaînes d'ensembles arbitraires (et mixtes) de caractères, pour autant que l'ensemble de caractères pour chaque chaîne soit identifié et qu'il soit possible de convertir la chaîne en Unicode. Les chaînes qui ont un ensemble de caractères non identifié et/ou ne peuvent pas être converties en Unicode ne sont pas rejetées, mais sont traitées comme binaires.

Chaque chaîne d'entrée est préparée en la convertissant en une chaîne "titlecased canonicalized UTF-8" *(UTF-8 canonisé en casse de titre)* en accord avec les étapes suivantes, en utilisant UnicodeData.txt [UNICODE] :

(1) Un codet Unicode est obtenu de la chaîne d'entrée.

(a) Si la chaîne d'entrée est dans un jeu de caractères connu qui peut être converti en Unicode, une séquence dans le jeu de caractères de la chaîne est lu et sa validité est vérifiée selon les règles de ce jeu de caractères. Si la séquence est valide, elle est convertie en un codet Unicode. Noter que pour les chaînes d'entrée en UTF-8, la séquence UTF-8 doit être valide en accord avec les règles de la [RFC3629] ; par exemple, les séquences UTF-8 trop longues sont invalides.

(b) Si la chaîne d'entrée est dans un jeu de caractères inconnu, ou si une séquence invalide se produit dans l'étape (1)(a), la conversion cesse. Aucune autre préparation n'est effectuée, tous les résultats partiels de préparation sont éliminés. La chaîne d'origine est utilisée inchangée avec le comparateur i;octet.

(2) Les étapes suivantes, utilisant UnicodeData.txt [UNICODE], sont effectuées sur le codet résultant de l'étape (1)(a).
(a) Si le codet a une propriété titlecase *(casse de titre)* dans UnicodeData.txt (c'est normalement la même que la propriété uppercase *(majuscule)*) le codet est converti en les codets dans la propriété titlecase.
(b) Si le codet résultant de (2)(a) a une propriété de décomposition de n'importe quel type dans UnicodeData.txt, le codet est converti en les codets dans la propriété de décomposition. Cette étape est appliquée de façon récurrente à chaque codet résultant jusqu'à ce qu'il n'y ait plus de décomposition possible (effectivement la forme de normalisation KD).

Exemple : le codet U+01C4 (LATIN CAPITAL LETTER DZ WITH CARON) a une propriété titlecase de U+01C5 (LATIN CAPITAL LETTER D WITH SMALL LETTER Z WITH CARON). Le codet U+01C5 a une propriété de décomposition de U+0044 (LATIN CAPITAL LETTER D) U+017E (LATIN SMALL LETTER Z WITH CARON). U+017E a une propriété de décomposition de U+007A (LATIN SMALL LETTER Z) U+030c (COMBINING CARON). Ni U+0044, U+007A, ni U+030C n'ont de propriété de décomposition. Donc, U+01C4 est converti en U+0044 U+007A U+030C par cette étape.

(3) Le ou les codets résultants de l'étape (2) sont ajoutés, en format UTF-8, à la chaîne "titlecased canonicalized UTF-8".

(4) Répéter à partir de l'étape (1) jusqu'à ce qu'il n'y ait plus de données dans la chaîne d'entrée.

Suite au processus de préparation ci-dessus sur chaque chaîne, les opérations d'égalité, d'ordre, et de sous chaînes sont comme pour i;octet.

Il est permis d'utiliser une autre mise en œuvre du processus de préparation ci-dessus si elle produit les mêmes résultats. Par exemple, il peut être plus pratique pour une mise en œuvre de convertir toutes les chaînes d'entrée en une séquence de valeurs UTF-16 ou UTF-32 avant d'effectuer les actions de l'étape (2). De même, si toutes les chaînes d'entrée sont Unicode (ou sont convertibles en Unicode) il est possible d'utiliser UTF-32 comme alternative à UTF-8 dans l'étape (3).

Note : UTF-16 ne convient pas comme solution de remplacement de UTF-8 dans l'étape (3), parce que les substituts de UTF-16 vont être cause que i;octet va apposer les codets U+E0000 à U+FFFF après des codets non BMP.

Cette collation n'est pas sensible aux éléments locaux. Par conséquent, il faut faire attention quand on utilise des fonctions fournies par le système d'exploitation pour mettre en œuvre cette collation. Des fonctions comme strcasecmp et toupper sont parfois sensibles aux éléments locaux et peuvent transposer des casses de lettres de façon incohérente.

La collation i;unicode-casemap convient bien pour être utilisée avec de nombreux protocoles de l'Internet et langages informatiques. L'utilisation avec un langage naturel est souvent inappropriée ; même si la collation prend apparamment en charge des langages comme le swahili et l'anglais, l'usage réel montre qu'elle tend à mal trier un certain nombre de types de chaînes :
- les noms de personnes et de lieux contenant des écritures qui ne sont pas assemblées selon "l'ordre alphabétique" ;
- les mots avec des caractères qui ont des signes diacritiques. Cependant, i;unicode-casemap fait généralement un meilleur travail que i;ascii-casemap pour la plupart des langages (mais pas tous). Par exemple, les lettres allemandes avec un umlaut sont triées correctement, mais certaines lettres scandinaves ne le sont pas ;
- les noms comme "Lloyd" (qui en gallois vient après "Lyon", à la différence de l'anglais) ;
- les chaînes qui contiennent d'autres symboles non lettres ; par exemple, les symboles Euro et Livre sterling, les marques de citation autres que "", les tirets/traits d'union, etc.

3. Enregistrement de collation Unicode Casemap

```
<?xml version='1.0'?>
<!DOCTYPE collation SYSTEM 'collationreg.dtd'>
<collation rfc="5051" scope="global" intendedUse="common">
<identifier>i;unicode-casemap</identifier>
<title>Unicode Casemap</title>
<operations>equality order substring</operations>
<specification>RFC 5051</specification>
<owner>IETF</owner>
<submitter>mrc@cac.washington.edu</submitter>
</collation>
```

4. Considérations sur la sécurité

Les considérations sur la sécurité pour les [RFC3629], [RFC3454], et [UNICODE-SEC] s'appliquent et sont normatives pour la présente spécification.

Les résultats de ce comparateur vont varier selon la mise en œuvre pour plusieurs raisons. Les mises en œuvre DOIVENT considérer si ces possibilités posent problème pour leur cas d'utilisation :

1) Les nouveaux caractères ajoutés dans Unicode peuvent avoir des propriétés de décomposition ou de casse de titre qui ne seront pas connues de la mise en œuvre sur la base d'une plus ancienne révision d'Unicode. Cela impacte l'étape (2).

2) L'étape (2)(b) définit un sous ensemble de la forme de normalisation KD (NFKD, *Normalization Form KD*) qui n'exige pas la normalisation des signes diacritiques déclassés. Cependant, une mise en œuvre PEUT utiliser un sous programme de bibliothèque NFKD qui fasse une telle normalisation. Cela impacte l'étape (2)(b) et éventuellement aussi l'étape (1)(a) et ce n'est un problème qu'avec les entrées UTF-8 mal formées.

3) L'ensemble de jeux de caractères traité dans l'étape (1)(a) est ouvert. UTF-8 (et, par extension, US-ASCII) sont les seuls jeux de caractères de mise en œuvre obligatoire. Cela impacte l'étape (1)(a).

   Les mises en œuvre DEVRAIENT, autant que faire se peut, prendre en charge tous les jeux de caractères qu'elles vont probablement rencontrer dans les données d'entrée, afin d'éviter une collation défectueuse causée par le repli sur la règle (1)(b).

4) Les autres jeux de caractères peuvent avoir des révisions qui ajoutent de nouveaux caractères qui ne sont pas connus d'une mise en œuvre fondée sur une révision plus ancienne. Cela impacte l'étape (1)(a) et éventuellement aussi l'étape (1)(b). Un attaquant peut créer une entrée mal formée ou dans un jeu de caractères inconnu, dans l'intention d'impacter le résultat de ce comparateur ou d'exploiter d'autres parties du système qui traitent ces entrées de différentes façons. Noter cependant que même des données bien formées dans un jeu de caractères connu peuvent impacter de façon inattendue le résultat de ce comparateur. Par exemple, un attaquant peut substituer à U+0041 (LATIN CAPITAL LETTER A) U+0391 (GREEK CAPITAL LETTER ALPHA) ou U+0410 (CYRILLIC CAPITAL LETTER A) dans l'intention de causer une non correspondance de chaînes qui paraissent visuellement les mêmes et/ou de causer l'apparition de la chaîne ailleurs dans un tri.

5. Considérations relatives à l'IANA

La collation "i;unicode-casemap" définie à la Section 2 a été ajoutée au registre des collations défini dans la [RFC4790].

6. Références normatives

[RFC3354] D. Eastlake 3rd, "Exigences du protocole du commerce ouvert sur Internet, version 2", août 2002. *(Information)*

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

[RFC4790] C. Newman et autres, "Registre de collation des protocoles d'application de l'Internet", mars 2007. *(P.S.)*

[UNICODE] <http://www.unicode.org/Public/UNIDATA/UnicodeData.txt>. Bien que le fichier UnicodeData.txt référencé ici fasse partie de la norme Unicode, il est sujet à des changements lorsque de nouveaux caractères sont ajoutés à Unicode et que des erreurs sont corrigées dans les révisions d'Unicode. Par suite, il peut être moins stable que ne le serait impliqué autrement par le statut de norme de cette spécification.

[UNICODE-SEC] Davis, M. and M. Suignard, "Unicode Security Considerations", février 2006, <http://www.unicode.org/reports/tr36/>.

7. Références pour information

[BASIC] Newman, C., Duerst, M., and A. Gulbrandsen, "i;basic - the Unicode Collation Algorithm", Travail en cours, mars 2007.

[RFC5256] M. Crispin, K. Murchison, "Protocole d'accès au message Internet - extensions SORT et THREAD", juin 2008. (*MàJ par* RFC5957) *(P.S.)*

Adresse de l'auteur

Mark R. Crispin
Networks and Distributed Computing
University of Washington
4545 15th Avenue NE
Seattle, WA 98105-4527
USA
téléphone : +1 (206) 543-5762
mél : MRC@CAC.Washington.EDU

Déclaration complète de droits de reproduction

L’IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d’adresser les informations à l’IETF à ietf-ipr@ietf.org.