Network Working Group
Internet-Draft
Expires: May 15, 2002

P. Faltstrom
Cisco Systems Inc
November 14, 2001

Lookup, Search and Sort
draft-faltstrom-i18n-sorting-00.txt

Status of this Memo

Copyright Notice

Abstract

Lookup, searching and sorting is somewhat defined when using a limited set of characters, and especially only one or a few scripts. When operating in a more complex environment, the algorithms get more complex, and in many cases need information about locale.

This document describes some of the problems which one encounter in an international environment.

1. Introduction

Characters mentioned in this document are identified by their position or code point in the Unicode character set [UCS]. The notation U+12AB, for example, indicates the character at the position 12AB (hexadecimal) in the [UCS]. It is strongly recommended that a [UCS] table is available for reference for the ideograph described.

In various applications the communication involves a requester and a servant of information. The requester ask for something, and the servant send back the result of the operation which takes place on the servant side of the communication link. The communication between the requester and the servant is done according to the definitions of the protocol which is used.

If the same request is coming from two different requesters, the result given back from the servant is supposed to be the same, or else the requester will be surprised. For example, if a requester is accessing the same servant from two different places in the world, the requester is supposed to get the same result back in both cases. Also, if two different requesters send the same request to the servant, they are supposed to get the same result back.

The request itself include some definition created by the requester on what information he wants from the servant. This request is in the case of textual based data expressed through a textual string, which of course can be encoded according to the transport encoding mechanisms defined in the protocol used.

This gives that the requester compose a request in his local environment, pass that request in the application protocol which is used to the servant, which is parsing the request, computes a result which is passed back to the requester.

The problem with lookups, searches and sorting is that the request is created in the local environment of the requester, and the computation which leads to the result set which is passed back from the servant is computed in the local environment of the servant. In a global environment these local environments might be different. Examples of differences are character sets, scripts, and locale dependent issues like language and geographical locality.

The environmental variables which can change the computational algorithm at the servent include: - Character set - Language - Geographical locality - Sort order definition(s) - Equality definition(s) - Case folding definition(s)

2. Lookup

A lookup is done when the requester know the exact key or value for the data which is requested. If the key is not exactly the right one, no data is returned. When using a distributed system, this implies also knowing what server to send the query to. In turn this implies that the key must include enough information to know where to send the query, and what the query should be (normally the key itself).

Example of a system which is using lookup is the Domain Name System.

2.1 Problems

match is a relativly simple operation, as the computation at the servent involves making a decision about what is "equal". This gives that the requester and the servant needs to agree on what is equal. Equality for text is defined by comparison of characters, and can be defined being case depenent or independent.

A classical example of equivalence definition problems inside one script can be illustrated using the character 'A' with a ring above. This character can be encoded in Unicode in three ways: 1) U+00C5 LATIN CAPITAL LETTER A WITH RING ABOVE 2) U+0041 LATIN CAPITAL LETTER A followed by U+030A COMBINING RING ABOVE 3) U+212B ANGSTROM SIGN The equivalence between 1) and 3) is a singleton equivalence. The equivalence between 1) and 2) is a precomposed/decomposed equivalence, where 1) is the precomposed representation, and 2) is the decomposed representation. In all three cases, it is supposed to look the same for the reader.

Case folding gives yet some other problems. In many parts of the world the character 'i' (U+0069, LATIN SMALL LETTER I) has as upper case equivalent the character 'I' (U+0049, LATIN CAPITAL LETTER I). That is not the case in Turkey. There they have a character U+0131 (LATIN SMALL LETTER DOTLESS I) which is lower case of U+0049 (LATIN CAPITAL LETTER I), and also a special upper case character which is the upper case equivalent of the character 'i'.

Multiple scripts introduce yet more complicated problems if the computation on the servant side is not done carefully. Basically, one can get matches inside any of the scripts (i.e. both the request and the data at the servant is in the same script) but also between scripts. This because some requests might be possible to express in more than one script (i.e. the same character might exist in more than one script).

Another issue have to do with "unification" of characters (or not)

between different languages and scripts. In Unicode a unification between Chinese, Japanese and Korean (CJK) characters were used, which means in practice that the "same" character used in any of the languages are unified to the same codepoint. This implies that given a character and no context, one doesn't know if one talk about the Chinese version of the character or the Japanese. In some environments knowing the difference is important for example when trying to define matching rules between the simplified and traditional form of the chinese character when used in the chinese language.

3. Sort

If more than one item is given back in the result set, there is a need to decide in what order the results are presented in the result set. The decision on ordering results normally consists on two steps: deciding what characters to use for creating the order and calculation of the order between two characters.

An example of the former is a decision that sorting of white pages records is to be done according to the last name of a person. The latter is the process of calculating the order between different last names.

Example of a system which allowes sorting is a white pages service using the LDAP protocol.

3.1 Problems

What characters to use when doing sorting changes between applications and between cultures. A phone book on Iceland is sorted on the first name of a person while one in Sweden is sorted on the last name for example. A list of email in an email application might be sorted in different ways (date, sender, subject) in the same session. What way to sort records is in many cases a request from the client to the server, and the server tries to fulfil that request just like it tries to fulfil the request of finding the data itself. This in many cases leads to a need for negotiation between client and server 031-934660. Gun Segerling

When comparing the individual characters with each other, the collation order is normally dependent on the language which is used, which means that cross language collation order is very hard to define (it is in most cases undefined, so there is no "standard" to reference). For example, the order of the local characters used in Danish, Swedish and Norwegian is different in the three countries, even though they in some way can be seen as being the same.

4. Search

When doing searches, compared with lookups, just matching characters is not enough as a computation algorithm. The data which is to be searched have to be structured in such a way that (more or less) the matching itself can be done in an arbitrary part of the data, and not only the key of the record. This is in many cases when the datastore is centralized (i.e. all data is stored at one servant) about the same problems as in the case of lookups, but if the data is distributed among several servants, finding the exact record is a non-trivial task. In the case of systems (such as DNS) which uses lookups, the structure of the key is normally chosen in a way so the key itself gives information about what servant have information about the record. Searches might also include getting information back from more than one servant if the data is distributed.

If the search term allow expressions like the regular expression "[a-c]" (match every character between 'a' and 'c', inclusive) the same problems occur like is described below regarding sorting, i.e. that a rank of each character is needed. The example given, LDAP, allow such expressions, so even though in LDAP a specific sort directive is not given, the computation which is taking place on the servant might in some cases involve sorting data.

4.1 Problems

The problems with searching is about the same problem as when doing a lookup, with the difference that a lookup is with an exact key, while a search is done via approximate matchings. Either approximate on the level of character by character, or one level above, as a semantic match ("color" and "colour").

Author's Address

Patrik Faltstrom
Cisco Systems Inc
170 W Tasman Drive SJ-13/2
San Jose CA 95134
USA

EMail: paf@cisco.com
URI: http://www.cisco.com

Appendix A. Appendix

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.