Sorting it all out: An introduction to collation

Cathy Wissink
Windows Globalization, Microsoft Corporation

Michael Kaplan
Trigeminal Software

1. Introduction.

A few months ago, the two of us were discussing an interesting bug in Windows that had to do with collation. After much discussion about both the code and the underlying linguistic phenomena that contributed to the bug, the comment was made that "sorting is a lot like alchemy". There is a great deal of truth to that comment—you call a seemingly insignificant function with a locale parameter, it references what appears to be a random set of numbers for your strings, and lo and behold, your strings are ordered in a fashion that makes sense to your users.   There is a great deal going on behind the scene that the user or developer doesn't need to worry about (that is, if the function is called correctly, you will get culturally-correct results), but the question still remains: what goes into the data behind a collation function? What kinds of linguistic phenomena impact collation?

Both of us get these kinds of questions quite often, along with the simpler questions of "well, what exactly is collation?" and "what do I need to know about collation"? As such, we felt there was a need for a presentation that addresses these concepts. The following paper will outline the basic concepts of collation, describe some of the basic linguistic and orthographic phenomena that influence sorting elements, debunk some myths about collation, and describe sorting best practices.   It is hoped that the reader will come away from this paper with a basic understanding of what influences culturally-correct collation, as well as an idea of where sorting plays a role in well-globalized software.

2. Basic concepts of collation.

What is collation? Sometimes called sorting, ordering or even "alphabetizing", collation is the culturally expected order of linguistic characters in a particular language[1]. In other words, speakers of a particular language have certain expectations of where to

[1] The Unicode Standard defines collation as: "The process of ordering units of textual information. Collation is usually specific to a particular language. Also known as alphabetizing or alphabetic sorting…" (http://www.unicode.org/glossary/)

find strings, relative to other strings, when in a collated (or sorted) list. Using an easy example here, in English, a speaker expects a word starting with Q to sort after all words beginning with P and before all words starting with R.

Outside of technology, people use linguistic sorting everyday:

- Searching for the name of an individual in a telephone book;
- Looking for a subject in a book index;
- Searching for a word in a dictionary;
- Using the (still frequently-used) library card catalog;

Basically, anytime a user orders data or searches for data in a logical fashion within any kind of structure, collation is being used. If the set of data is sorted correctly for the language, the user will find the needed data quickly and efficiently. Conversely, if the list is sorted in some fashion that is not expected by the user, it will take more time and effort to find items in the list[2].

An interesting aspect of linguistic sorting is the speaker's (generally) subconscious knowledge of it. Native speakers have clear expectations of where to find data in a collated list; that is, they can easily identify a sorted list as correct or incorrect for their culture. In addition, users can generally describe the simple qualities of their collation; for example, most users can easily describe how the basic linguistic characters (e.g., "letters"[3]) of their language sort. However, when it comes to more complicated phenomena, such as "accented" characters (diacritics, matras), interaction with punctuation (e.g., hyphens), or compressions (e.g., the Spanish CH), it is often harder for users to explain how a sort works[4].

3. Features of a language and their influence on collation.

While the speaker of a language may not have conscious or overt knowledge of the different phenomena that influence linguistic sorting for his or her culture, these

[2] A (true!) example of this is in Cathy's office building at Microsoft. For some odd reason, office mail slots are not sorted by name, but by an odd numbering scheme involving building floor and office. Users expect to find a mail slot by name, and even after using this collation system for some time, we find it difficult and frustrating (since, after all, who can ever remember where someone's office sorts relative to all other offices in the building?).
[3] The term "letters" is chosen intentionally. To be technical, what is meant here are the linguistic characters that carry a primary collation weight; this could be ideographic, syllabic, or alphabetic in nature.
[4] This phenomenon is what makes the research of collation data so very difficult for many languages.

phenomena do indeed exist and must be taken into account when creating a collation. Some examples of linguistic elements that influence sorting include:

- Linguistic "characters"[5];
- Casing (upper vs. lower);
- Modifiers (diacritics, Indic matras, vowel marks)

A brief explanation of the key features follows[6]. (There are many other elements that impact collation on a lesser level, but that is a more advanced topic.)

<u>3.1 Linguistic "characters".</u> Linguistic characters are those characters that, for lack of a better term, a speaker recognizes as belonging to the language. (Technically, this is defined as a grapheme[7], but for simplicity's sake, we will use the term linguistic character.) It is important to note that not all linguistic characters are a single code point in an encoding (or in Unicode), as the following European examples demonstrate:

- Spanish CH: consists of two code points C (U+0043) and D (U+0044) and sorts as a unique character between C and D[8];
- Hungarian DZ: sorts as a unique character between D (U+0044) and DZS (note that DZS is also a unique character for Hungarian in its own right, sorting before E and after DZ);
- Norwegian Ø (U+00D8): sorts as a unique character after Z (U+005A) and Ä (U+00C4). (This particular character could also be rendered as a combination of O [U+004F] and a non-spacing slash [U+0338].)

Speakers of a language expect words (or strings) in a list to be grouped according to the linguistic characters of their language[9]. There may be some variations on these characters (for example, a letter is upper or lower case, or has a diacritic modifying the

---

[5] See footnote 3 for more clarification.

[6] For clarity's sake, these definitions are neither all encompassing nor completely accurate from a linguistic perspective. These definitions are meant to assist the <u>novice</u> in understanding the basic elements of a writing system that impact collation. People with a greater understanding of either collation or writing systems will find these definitions somewhat simplistic.

[7] For yet another definition of grapheme, here is the entry from the Unicode glossary: "*Grapheme*. (1) A minimally distinctive unit of writing in the context of a particular writing system. For example, *b* and *d* are distinct graphemes in English writing systems because there exist distinct words like big and dig. Conversely, a lowercase italiform letter *a* and a lowercase Roman letter a are not distinct graphemes because no word is distinguished on the basis of these two different forms. A grapheme is for a writing system what a phoneme is for a phonology. (2) What a user thinks of as a character."

[8] This is the older Spanish "Traditional" sort. The modern sort does not use this character.

[9] For a more technical take on this concept, please see Cathy's paper *Issues in Indic Collation* at http://www.unicode.org/notes/tn1/.

character), but these are the core characters by which words in a language are ordered. To further clarify the concept of these core characters, see the following list:

apple
Apple
Are
ban
banana
Banana

Notice that there is a significant enough difference in sorting between the letters A and B, such that all the strings beginning with A come before all the strings beginning with B. This difference is referred to in collation as weight. The weight of linguistic characters takes precedence over all other weights we will discuss later in the paper. In addition, the weight of a linguistic character is often called a primary weight[10].

It is also important to note that what determines a linguistic character (and respectively a collation primary weight) depends on the writing system and the specific language. In alphabets, the linguistic character is generally a letter. In ideographic languages (e.g., Chinese, Japanese, Korean), the linguistic character is determined by a number of factors far beyond the scope of this paper; suffice it to say that primary weighting for the linguistic characters in the various languages can be based on the main radical within the character, the stroke count of the character, or even pronunciation (e.g., Taiwanese "Bopomofo"). Some languages with inherent vowels (e.g., Devanagari-script languages, Tamil) will have particular modifiers to determine linguistic characters and primary weights[11].

(We will show more examples of these linguistic characters later, when we discuss specific examples of collation in different languages.)

3.2. Casing. Casing is that aspect in certain writing systems (e.g., Latin, Cyrillic, Greek), where there are two forms of a letter: a big (or uppercase) variant, and a small (or lowercase) variant. For example, there are two variants of Latin letter P (U+0050 or "P", and U+0070 or "p"), and there are two variants of Cyrillic letter Pe (U+041F or "П", and U+043F or "п").

Casing does impact sorting, but to a lesser extent than the linguistic characters of a language do; the weighting of case has less of an influence than linguistic characters. In

[10] The concept of weighting and sort keys is far beyond the scope of this paper. For a good explanation of this concept, please see the Unicode Collation Algorithm (http://www.unicode.org/unicode/reports/tr10/).
[11] Note that this is not the case in all Indic languages.

other words, while the linguistic characters of a language group words into collections within a list, casing influences how a character may sort relative to a different version of that character.

To contrast the concepts of linguistic characters vs. casing in sorting, let's go back to the sorting example given a few paragraphs earlier. As noted earlier, the primary weight of the linguistic characters creates a grouping of "A" words, followed by "B" words.

Now compare the difference between "apple" and "Apple". The words are identical, except for their case. From a collation perspective, you need to give the concept of case a weight, such that one comes before the other[12]. However, you do not want to weight the case of the letter A so much that it ends up being ordered with the B words. The weight of case (of the first letter of the string) therefore needs to be less than the weight of character.

3.3. Modifiers. Modifiers are exactly that—those elements within a writing system that modify linguistic characters. You often see them somewhere around a linguistic character (e.g., on top, underneath, to the side). Some examples include:

The acute sign in Latin e acute (É);
The tonos sign in Greek epsilon tonos (έ);
The dagesh in Hebrew Fay (or Pay + Dagesh) (פ);
The nukta in Devanagari Qa (or Ka + nukta) (क़).

As you have perhaps ascertained by now, modifiers also impact ordering in languages. Often, they will have a lesser weight, not unlike case. For example, it could be the case that in a Latin-based language, diacritics are sorted in the following manner: acute < grave < diaeresis.

This language would, as a result, sort characters with diacritics in the following order:

á (Lower Case A Acute)
Á (Upper Case A Acute)
à (Lower Case A Grave)
À (Upper Case A Grave)
ä (Lower Case A Diaeresis)
Ä (Upper Case A Diaeresis)

[12] There is much discussion about which variant of case comes first: lower or upper case. For this particular presentation, we will assume ordering of lowercase characters before uppercase characters.

Most of the time, this modifier information is considered more important than case but less important than the base character[13]. The reason for this is simple – in most instances, people expect the "accented" versions of a letter to be sorted in the same way independent of their case.

Another way in which you can see the less distinct role of case is in the frequent use of case-insensitive searching and sorting, compared to modifier-insensitive ("diacritic insensitive") searching and sorting. Two characters that only differ by case are considered by most users to be the "same" character, whereas that cannot always be said for two characters that differ only by modifier. (This of course does depend on the language.)

Now that you have a basic idea of how some linguistic elements might impact collation, let's look at some language examples to put these concepts into practice.

4. Collation in action: examples of culturally-correct collation.

The easiest way to explain the inner workings of sorting is to show some examples of different linguistic collations applied to the same data.

We created a simple list of multilingual strings, and then applied English collation rules. The results are shown below. (The numbers to the left are included so that when the data is re-ordered for different collations, as seen later in the document, differences will be easier to determine.) This particular set of words was arbitrarily chosen to help highlight some of the differences one can expect to see across a number of different collations.

Figure 1: English collation order.

1	casa
2	chaque
3	choisi
4	choy
5	císta
6	Ciuriką
7	cote
8	coté
9	côte
10	côté
11	cycle
12	hier

[13] In other words, the modifier weight is greater than the case weight, but less than the linguistic character weight.

13	hören
14	hybrid
15	Ivo
16	Jahr
17	jamais
18	Jūs
19	leaba
20	libros
21	lösen
22	lietuvių
23	llueve
24	luath
25	lycée
26	lyti
27	Yvonne

Note that basic A to Z order is used in the above English example. All accented linguistic characters sort with their base characters; diacritics affect the string from left to right.

That same list of strings, sorted according to French collation rules is shown below:

Figure 2: French collation order.

1	casa
2	chaque
3	choisi
4	choy
5	císta
6	Ciuriką
7	**cote**
9	**côte**
8	**coté**
10	**côté**
11	cycle
12	hier
13	hören
14	hybrid
15	Ivo
16	Jahr
17	jamais
18	Jūs
19	leaba
20	libros
22	lietuvių
23	llueve
21	lösen
24	luath
25	lycée

26	lyti
27	Yvonne

The major difference you will see in French collation (as compared to English collation) is the treatment of diacritics (accented characters). Note that in the English sort, diacritics impact the weighting of a string from left to right, that is:

7	cote
8	coté
9	côte
10	côté

In particular, notice how the two strings beginning with "co-" come before the strings starting with "cô-". The diacritic weight of the circumflex (the accent on items 9 and 10) impacts the letter O enough to move these two items below the unaccented O items.
In comparison, in French, diacritics impact the weighting of a string from right to left, that is:

7	cote
9	côte
8	coté
10	côté

Notice how the strings ending with "-e" come before strings ending with "-é".   The diacritic weight of the acute (the accent on items 8 and 10) impacts the letter E enough to move these two items below the unaccented E items.

In Swedish or Finnish, the order is yet again different:

Figure 3: Swedish/Finnish collation order.

1	casa
2	chaque
3	choisi
4	choy
5	císta
6	Ciuriką
7	cote
8	coté
9	côte
10	côté
11	cycle
12	hier
14	**hybrid**
13	**hören**
15	Ivo
16	Jahr

17	jamais
18	Jūs
19	leaba
20	libros
22	lietuvių
23	llueve
24	luath
25	lycée
26	**lyti**
21	**lösen**
27	Yvonne

While there are other differences that make Swedish and Finnish sorting unique (that are not discussed here), the major distinction that stands out in this example is the different treatment of the o diaeresis/umlaut (Ö). Note that here, it is a unique character that sorts after z, as shown in the following examples:

12	hier
14	**hybrid**
13	**hören**

24	luath
25	lycée
26	**lyti**
21	**lösen**

In Swedish and Finnish, the Ö carries a letter weight that is heavier than Z. While the O is accented with a diaeresis, it is not the same as an O + a diacritic weight; it is a unique character in its own right.

Comparing the earlier English and French examples, o diaeresis was considered a variant of o, that is:

12	hier
13	**hören**
14	**hybrid**

21	**lösen**
24	luath
25	lycée
26	**lyti**

For Traditional Spanish, which has fallen out of common usage with the advent of the Modern Spanish collation, the differences are even more extensive. While this older style of sort is not used much, it is an interesting example of multiple letters working as a single unit (e.g., a single linguistic character) in collation.

Figure 4: Traditional Spanish collation order.

1	casa
5	císta
6	Ciuriką
7	cote
8	coté
9	côte
10	côté
11	cycle
2	**Chaque**
3	**choisi**
4	**choy**
12	hier
13	hören
14	Hybrid
15	Ivo
16	Jahr
17	jamais
18	Jūs
19	leaba
20	libros
22	lietuvių
21	lösen
24	luath
25	lycée
26	lyti
23	**llueve**
27	Yvonne

In Traditional Spanish, CH and LL are treated as unique characters that come between C and D, and L and M respectively. CH was discussed earlier in section 3.1; LL works in a similar manner, in that all strings beginning with L (but not LL) must be exhausted before going on to all strings that begin with LL. Compare the English treatment of CH and LL in the two examples below:

1	casa
2	**chaque**
3	**choisi**
4	**choy**
5	císta

22	lietuvių
23	**llueve**
24	luath
25	lycée
26	lyti

In English (and many other Latin script languages), strings beginning with CH sort after strings starting with CG and before strings beginning with CI; in other words, all strings are compared on C before moving to the next letter. (There is an analogous example with LL.) In Traditional Spanish, however, CH and LL must be treated as elements onto themselves, and as such, they sort in a very different manner, as shown below:

9	côte
10	côté
11	cycle
2	**Chaque**
3	**choisi**
4	**choy**

21	lösen
24	luath
25	lycée
26	lyti
23	**llueve**

As demonstrated by the two above examples, Traditional Spanish uses CH and LL as unique "compressions" (that is, multiple Unicode code points sorting as a single linguistic character).

Perhaps one of the most visually striking collations to users familiar with many variants of sorting in Latin script languages, but unfamiliar with this particular language is the Lithuanian collation:

Figure 5: Lithuanian collation order.

1	casa
2	chaque
3	choisi
4	choy
5	císta
6	Ciuriką
11	**cycle**
7	cote
8	coté
9	côte
10	côté
12	hier
14	**hybrid**
13	hören
15	Ivo
27	**Yvonne**
16	Jahr
17	jamais
18	Jūs

19	leaba
20	libros
22	lietuvių
25	**lycée**
26	**lyti**
23	llueve
21	lösen
24	luath

Note that within Lithuanian, the letter Y sorts as a unique letter immediately after I, as seen in the following examples:

5	císta
6	Ciuriką
11	**cycle**
7	cote

Comparing this result with the other sorts we've examined up to this point, see that Y sorts between X and Z, that is:

8	coté
9	côte
10	côté
11	**cycle**

As you can see by the European-language examples shown above, cultural expectations can vary a great deal, even within the same basic writing system, and within the same part of the world.

Let's now look at an East Asian language example, specifically Chinese in Taiwan. Developing an accurate sort order is arguably even more important in East Asia, due to the different structure of the two types of writing systems (our previous European examples come from alphabets; East Asian languages use ideographic systems). This is due to the magnitude (and difficulty) of grouping ideographs: if someone in Europe finds it challenging to randomly order 26-40 characters, the person in Taiwan who has over 2,000 ideographs to deal with has a much larger issue if there is no predictable ordering.

There are several different collation methods that can be used for Chinese in Taiwan; for example, Traditional Chinese ideographs can be sorted in Unicode code point order, or in the order of an encoding (that is, with no linguistic meaning), Bopomofo (pronunciation), or stroke count. Looking at a list of languages in both Bofomofo and stroke count orders, you will see a significant difference:

Figure 6: Chinese (Taiwan) sorting: Bopomofo vs. Stroke Count ordering.

Bopomofo	
巴斯克語	Basque
白俄羅斯語	Byelorussian
保加利亞語	Bulgarian
波蘭語	Polish
波斯尼亞語	Bosnian
波斯語	Farsi
葡萄牙語	Portuguese
馬來語	Malaysian
馬其頓語	Macedonian
法語	French
芬蘭語	Finnish
德文希伯來語	Yiddish
德語	German
丹麥語	Danish
塔米語	Tamil
塔特語	Tatar
塔吉克語	Tajik
塔吉克語（拉丁語）	Tajik
塔加拉族語	Tagalog
泰語	Thai
土耳其語	Turkish
南非語	Afrikaans
挪威語	Norwegian
拉脫維亞語	Latvian
立陶宛語	Lithuanian

Stroke Count	
土耳其語	Turkish
丹麥語	Danish
巴斯克語	Basque
白俄羅斯語	Byelorussian
立陶宛語	Lithuanian
拉脫維亞語	Latvian
波斯尼亞語	Bosnian
波斯語	Farsi
波蘭語	Polish
法語	French
芬蘭語	Finnish
保加利亞語	Bulgarian
南非語	Afrikaans
挪威語	Norwegian
泰語	Thai
馬來語	Malaysian
馬其頓語	Macedonian
塔加拉族語	Tagalog
塔吉克語	Tajik
塔吉克語（拉丁語）	Tajik
塔米語	Tamil
塔特語	Tatar
葡萄牙語	Portuguese
德文希伯來語	Yiddish
德語	German

In the Bopomofo order, strings are sorted by pronunciation (the term “Bopomofo” is the common name for phonetic transcription used in Taiwan; Bopomofo comes from the first 4 letters of this phonetic order; the technical name is *Zhuyin fuhao*). With the stroke count order, strings are sorted by number of strokes within the ideograph. If you compare the two lists above, looking at the equivalent English names of each string, you can see there is a significant difference between the two orders.

Finally, we'd like to point out some differences within Indic languages, specifically, Devanagari script and how sorting changes between two languages in this script: Hindi vs. Marathi. The Devanagari script is an example of yet another type of writing system, specifically, an alphasyllabary[14]. (Note that yet again, the type of writing system that a language uses does impact how sorting works for that particular language. You may find collation similarities across languages within the same type of writing system, but with each distinct type of writing system, it is important to determine the characters of the language and how they relate to code points in the repertoire.)

The example below shows how the code point U+0933 (Devanagari Lla) changes ordering, depending on the language.

Figure 7: Hindi vs. Marathi ordering.

Hindi:			**Marathi:**		
ल	Devanagari La	U+0932	ल	Devanagari La	U+0932
ळ	***Devanagari Lla***	***U+0933***	व	Devanagari Va	U+0935
ऴ	Devanagari Llla	U+0934	श	Devanagari Sha	U+0936
व	Devanagari Va	U+0935	ष	Devanagari Ssa	U+0937
श	Devanagari Sha	U+0936	स	Devanagari Sa	U+0938
ष	Devanagari Ssa	U+0937	ह	Devanagari Ha	U+0939
स	Devanagari Sa	U+0938	ळ	***Devanagari Lla***	***U+0933***
ह	Devanagari Ha	U+0939	क्ष	***Devanagari Ksha****	***U+0915, U+094d, U+0937***
			ज्ञ	***Devanagari Jnya*****	***U+091c, U+094d, U+091e***

**considered a conjunct in Hindi, but the 35th consonant in Marathi*
***considered a conjunct in Hindi, but the 36th consonant in Marathi*

As you have certainly ascertained by now, collation can vary, depending on a number of factors, and can be influenced by a number of elements.

5. *Myths about collation.*

[14] This is the definition given by Daniels and Bright (*The World's Writing Systems, 1996*, page 384). An alphasyllabary is a writing system that gives each consonant + vowel combination a syllabic quality; the vowel modifies the syllable.

After reading the previous two sections, you now (ideally) have an idea of the inner workings of collation and the practical application of these concepts to various languages. It is, however, still necessary to debunk some of the untrue statements that we've heard about collation over the last few years. While these myths may be obvious to you, we still hear them from customers, developers and users quite a bit, and you may hear them as well (since there are many misconceptions of what software globalization entails, especially at higher management levels where the technical difficulties may not be well-understood or investigated).

5.1. Misconception #1: "If I localize my product, I don't need to worry about details like collation".

This misconception shows a lack of understanding of the difference between localization and globalization. While localization clearly helps prepare a product to be used in a particular market, it is not sufficient to translate strings—the product must still be enabled for a local experience, including the ability to search and sort strings in a manner appropriate for the culture. As such, it is necessary to call (or write) functions that use culturally-correct collation.

5.2. Misconception #2: "If I use Unicode in my product, sorting is already covered".

This particular misconception is widespread, not only regarding Unicode, but also encodings in general (that is, there is a common belief that encoding order is a satisfactory collation order). Unicode clearly is the heart of a well-globalized product (for reasons too numerous to list here). That said, Unicode is a character encoding, not a means of collation[15]. While there are some code points that are in a reasonable order within Unicode, the default order given by Unicode is in no way an acceptable collation for any culture.[16]

5.3. Misconception #3: "One collation is good enough for Europe (North America, Asia), right?"

[15] Of course, the Unicode Collation Algorithm (http://www.unicode.org/unicode/reports/tr10/) is available as a Unicode Technical Standard. Note however that the UCA is defined as a specification independent of the Unicode Standard; conformance to the Unicode Standard does not imply conformance to the UCA.

[16] The document *Issues in Indic Collation* (see reference in footnote 9) goes into this general misconception in much more detail: "…collation for any language has certain structural needs that cannot be met by an encoding, and Unicode as an encoding is no exception. However, it should also be clear that character encodings were not developed to work as acceptable sorting orders; encodings are for the sole purpose of providing a relationship between a linguistic character and a number."

As shown by the examples in section 4, there are great differences in collation results depending on the language, even within the same geographical region. Therefore, it is highly unlikely that a "one sort fits all" approach would be satisfactory for all users within that particular part of the world. In addition, multiple writing systems are in use within particular geographical regions, and the approach taken for one writing system may be in no way sufficient for another writing system.

5.4. Misconception #4: "One collation is good enough for the Latin (Han, Cyrillic) script, right?"

This misconception is the same as #3, expanded to include more users. While there may be some writing systems that are used for only one language, and as such, one sort may be sufficient, this is the exception. Again, a "one sort fits all" approach would not be satisfactory to most users.

6. Keeping it under the covers.

Collation is something that, ideally, people will not notice when it is working well. Whether they are looking at the WAB (Windows Address Book) in Microsoft's Outlook Express or the files in a Windows Explorer listview, users will not think twice about the order if it is culturally correct. If it is wrong, however, they will definitely notice the problem.

You can actually see how radically collation can change in the user experience by changing your user locale (for example, on Windows through Regional Options in the Control Panel) to a locale that sorts differently than your ordering expectations. Even doing this for short periods of time will highlight how strange the system seems when it is "out of order."

Windows does tend to simplify this type of issue by supplying APIs like lstrcmp:

```
int lstrcmp(
  LPCTSTR lpString1,  // first string
  LPCTSTR lpString2   // second string
);
```

This function takes two null-terminated strings and returns -1, 0, or 1 depending on whether the first string would sort before, identical with, or after the second string. The comparison follows the user's cultural preferences, matching their chosen user locale. Similarly, there is an lstrcmpi function:

```
int lstrcmpi(
  LPCTSTR lpString1,  // first string
  LPCTSTR lpString2   // second string
);
```

The lstrcmpi function exists to do case insensitive comparisons of two strings and is otherwise identical to lstrcmp. APIs such as these allow a developer to do standard copying operations without having to worry too much about the larger issues of collation.

Sometimes simple functions like lstrcmp and lstrcmpi are not enough, and a more detailed or customized string comparison function is needed. For this, Windows provides functions that take a locale parameter such as CompareString. Additionally CompareString lets you alter other collation behavior when you need to. In many cases, however, lstrcmp and lstrcmpi can do everything you need.

6.1. Getting users the results they expect.

Now that you know how important it is to always follow the user's preferences, there are of course exceptions to this rule where the user might specifically expect there to be an ordering that does not match their preferences:

- In a file system, one might need to compare two file names in a case insensitive manner. This means that any two file names that differ only by case are invalid; they would conflict. Because sorting (and respectively casing) rules could vary from user to user, you could end up with files that are valid on one user's machine and invalid on another. This is a bad thing.
- Looking at applications like Excel, the columns are given letters. These letters are used in combination with the numbered rows so that cells can be identified as A1 or B12 or FD45. If one changed the ordering, then VBA code and formulas in the spreadsheet would break.
- In this multilingual and multicultural society, your user might be Swedish but they may be looking at German data. The user's expectations for collation are not based on the data, but (generally) his or her native language.

In the cases where one would not want to use a linguistically appropriate collation, the desire to be *too* linguistically appropriate can actually be a source of confusion, errors and more problems than you can shake a stick at. Certainly one of the biggest reasons why for example a file system cannot support Turkish casing rules[17] is due to the fact

[17] In Turkish (and Azeri Latin), dotless I (U+0131 and the upper case variant) sorts as a unique linguistic character before dotted I. (U+0130 and the lower case variant) In most other Latin languages, lowercase dotted I sorts as a cased variant of uppercase dotless I (uppercase dotted I and lowercase dotless I are not used).

that .GIF, .Gif, and .gif files all need to be seen as the same file type so that a graphics program can find the file.

Obviously there is no one simple rule to be followed, given how much variation there is. The designers of the application simply need to decide what the users will be expecting, and whether it is necessary to have consistent (and potentially linguistically inaccurate) results, and if these two concepts overlap at all.

Note that there are two separate issues here: expecting consistent results independent of language, and giving users a culturally-correct experience. In the former case, one wants an invariant collation, and in the latter, one wants to follow the user's expectations. For example, the comparison of two files (for the sake of absolute equality) cannot change from machine to machine, the ordering of those files when one is looking at a directory on their own machine usually can be culturally valid without causing any problems.

7. Summary.

Collation is an integral part of a well-globalized product. As described in section 6, it is just pervasive in software as it is out in the real world, and as such, it is necessary to provide users with a means to search and order data in a way that makes sense in their particular culture. Unfortunately, there is no good way to cut corners when it comes to implementing collation and group sorts by region or writing system; it is necessary to take each individual language into account in order to get culturally accepted results[18]. That said, there are many tools (e.g. ICU, the C runtime and its comparison functions) and APIs (e.g. lstrcmp, lstrcmpi, and CompareString) out there to help developers and designers plan and implement the well-globalized product, and the most important thing one can do is to stop and consider what the user might be expecting here prior to doing a lot of work that will confuse them. When collation is implemented thoughtfully and with prior consideration for the users, it can be a powerful force in making the software easier to use.

[18] You may find that there are some linguistic sorts that are either identical across languages, or are perhaps a subset of another collation; however, this is rarely the case, and one should assume that collation is always on a per-language basis.