Standards

Toward Decent Text Encoding

Neville Holmes, University of Tasmania

Text is composed of characters; we get different kinds of text from different kinds of characters. So character sets are very important. And if there are contending views about whether we are well-served by our character-set standards, these views should be exposed and discussed.

It's strange that the computing industry has for so long stuck to poor and impoverished character sets for text encoding. Now, without much public discussion or dispute, the computing industry seems to be moving to an equally poor but contrastingly obese character set called Unicode.

It's strange that the computing industry has stuck to poor and impoverished character sets for text encoding.

TRADITIONAL CHARACTER SETS

The development of writing technology—and, relatively recently, of print technology—has been more a story of the gradual development of standards than a story of the development of machinery. The widespread acceptance of the roman and italic forms of the Latin alphabet—which have become the dominant alphabetic forms in countries such as Germany and Turkey only within living memory—has added an important interlingual aspect to the use of character sets and to international use of print technology.

The early development of automatic data processing mainly in English-speaking countries led to English versions of the Latin alphabet being used in associated machinery, particularly in printers. In the 1950s, the typical line printer sported 26 letters (uppercase roman), 10 decimal digits, and a few special characters added mainly for commercial use. If Fortran programmers wanted to see their additions normally then they had to order their machines with a special feature to replace the ampersands with + signs. And they were forced to use the asterisk as a multiplication symbol.

In the 1960s, two expanded character sets came into wide use. When IBM introduced the 8-bit System/360 computers, it introduced an 8-bit character set called EBCDIC (Extended Binary Coded Decimal Interchange Code) to go with it. A particular desire for compatibility with prior punched-card codings gave a quite bizarre structure to this de facto standard.

At the same time, a formal effort resulted in a 7-bit standard character set called ASCII (American Standard Code for Information Interchange), which was particularly designed for the telegraphy of the time.

Editor: Charles Severance, Michigan State University, Department of Computer Science, 1338 Engineering Bldg., East Lansing, MI 48824; voice (517) 353-2268; fax (517) 355-7516; crs@egr.msu.edu; http://www.egr.msu.edu/~crs

EBCDIC AND ASCII

Both EBCDIC and ASCII were put together with a great deal of thoughtful effort and are still widely used. They included upper– and lowercase Latin alphabets, the 10 decimal digits, a slightly enhanced but still inadequate set of special characters, and a set of noncharacters intended to be used for controlling recording machinery in various ways.

Having two character sets in concurrent widespread use has been problem enough, but there have been many other problems. Both EBCDIC and ASCII provided users with a + symbol as standard, but (with breathtaking arrogance) the developers of both sets refused to provide the traditional multiplication and division symbols. The control characters not only proved inadequate but were used inconsistently. For example, both Unix and IBM PC operating systems have traditionally used the ASCII character set, but in encoding text Unix has used a single line feed control character to separate lines of text, while IBM PCs have used a carrier return/line feed control character pair.

Both character sets became grossly distorted when they were adapted to encoding text in languages other than English. ASCII had a problem anyway in being a 7-bit coding typically running on 8-bit machinery, which led to peculiar and inconsistent uses of the eighth bit. But the way in which versions of ASCII were accreted for new languages was grotesque in the extreme.

UNICODE CHARACTER SET

Little wonder, therefore, that the computing industry should wish to replace EBCDIC and ASCII with a new improved character set called Unicode, particularly when computing has become so international. What is amazing is that it has taken this long. What is disappointing, if not tragic, is that the replacement is so unsuitable for text encoding.

There has been relatively little popular discussion of Unicode. A recent exception is the complementary proposal by Muhammad Mudawwar ("Multicode: A Truly Multilingual Approach to Text Encoding," *Computer*, Apr. 1997, pp. 37-43). Unicode seems to be trying to provide a single character set to represent documents in any language or writing system or mixture thereof. A large part of the difficulty with Unicode, though, is that it is most suitable for—even aimed at—presenting text, not for encoding it. But presentation of text is one technology, while the encoding, storage, and transmission of text is quite another.

Unicode is intended primarily to allow the computing and telecommunications industry to get by with only one character set for the entire world (http://www.unicode.org). One result is that everyone has to use 16 bits for every character. Surely it would be sadistic to suggest that the great redundancy involved in a 16-bit character set would allow the effective use of data compression techniques. Or to suggest that everyone's equipment should support all the world's writing systems, past and present, at the same time. But with Unicode it's either that or back to proliferating versions.

Mudawwar's Multicode aims to counter the 16-bit drawback and several others that he describes in some detail. But Multicode is essentially a compromise; Mudawwar's article emphasizes in its very last sentence that "both approaches can coexist—Multicode for programming ease and Unicode to support unified fonts." But in international communication, the necessary variety of Multicodes would be much more complex than the single Unicode.

ENCODING TEXT

Most traffic in text is raw text—messages, identifiers, business records—and the vast majority of this traffic is monolingual. Indeed, the vast majority of presented documents are also monolingual. Much of this monolingual text needs only an 8-bit encoding system to be encoded as plaintext.

Mudawwar's Multicode scheme recognizes this and therefore provides for a separate character set for every "official language" ("Unicode Misunderstood," *Computer*, June 1997). Most of these languages can be accommodated within an 8-bit coding scheme. In this case Multicode provides for great data compression, but in any case it separates languages from one another, which is no longer the way of the world, if it ever was.

There are two aspects of language interchange. First, languages borrow words and phrases from one another so that, for example, English uses French and German words and takes their diacritical marks with them. Readers are often better served when the markings are kept as they are, showing how words like "café" and "cliché" should be pronounced.

Second, in this international society it is important to be able to name people and organizations in their own language. Indeed, to many cultures it is insulting not to use their names properly. In Western text, Chinese names are stripped of the tone marks provided in their Pīnyīn spelling system, which would be equivalent to English usage stripping French or German names of all their vowels. Most unfriendly behavior.

For text encoding, the world needs a standard for each writing system that suits each and every language using that system.

The world is divided into writing system zones. For languages that use the same writing system—the system based on the Latin alphabet, for example—a good text-encoding standard would completely support the exchange of names. I should be able to read all Swedish names in plaintext e-mail messages, but at present many are garbled. On the other hand, writing system cultural zones expect to transliterate words and names from other zones, which seems to be a quite amenable approach, provided it can be done well.

NECESSARY STANDARDS

For text encoding, the world needs a standard for each writing system that suits each and every language using that system. These standards should be in accord with each other so that basic processing—such as distinguishing letters from punctuation and numerals from words—is the same for each system and also so that text using one writing system can be practically encoded or viewed on equipment designed for another writing system.

Text encoded by these systems could be marked up for presentation within the writing system that encodes it. Using markup would surely provide a logical, effective, and efficient separation of function and would make it easy to combine text from different writing systems.

Most of the world's writing systems could probably be encoded using an 8-bit scheme. The one exception is the traditional Chinese writing system, which encompasses thousands of distinct characters. But it could be argued that this rich and time-honored character system is more properly regarded as a reading system because it is more efficient for reading than the various alphabetic systems but it is much less effective for writing/encoding.

There is evidence that languages that use Chinese characters could be encoded under an 8-bit scheme. For example, two articles in *Computer*'s last special issue on such matters (Jan. 1985) proposed encoding the Chinese official spoken language, Pūtōnghuà, using its Pīnyīn alphabetic system. As recently as last year, an 8-bit encoding system has been introduced in South Korea in which the Korean alphabet is used to encode the Chinese characters they use. This system is being adopted in many circles.

There is some reason to hope that a family of 8-bit text encoding standards could be designed to suit all the world's writing systems, a family that would provide for cheap, efficient, and effective machinery for encoding, storing, and transmitting the world's text. The most important gain from adopting text encoding standards for each writing system (concordant between writing systems) is that simple and effective equipment, and text processing software such as plaintext editors, e-mailers, and Internet search engines, could be largely independent of the language within any writing system.

Also, text could be compatibly handled across writing systems by such software even if the equipment wouldn't display or print the text "correctly." Words would still look like words, numerals like numerals, and punctuation like punctuation. ❖

Neville Holmes is a senior lecturer in the School of Computing at the University of Tasmania. Contact him at neville.holmes@utas.edu.au.