Philology

Character Set Encoding

Nelson H.F. Beebe

Introduction

The article by Janusz S. Bień [3] which follows this paper complements an earlier one by Yannis Haralambous [5] on the subject of support for larger character sets.

Because this is an area of international interest in the computing community, it seemed worthwhile to review some of the issues, in order to provide background for those readers who are not actively following the subject.

There are currently at least two ISO groups that are actively engaged in the standardization of character set encoding. They are identified here by the reference numbers of the standards on which they are working, ISO 8859 and ISO 10646.

The ISO 8859 group deals with ASCII and EBCDIC character set issues and with standardization of 8-bit character sets. The ISO 10646 group deals with multi-byte character set issues.

I have been following the ISO 8859 work for more than two years, and recently joined the ISO 10646 discussions. Based on that experience, it seems clear that the problem is much more difficult than most people realize.

Both groups have active electronic mailing lists; the end of this article has information on how to subscribe to them.

256 Does Not Suffice

There is a need for more than 256 characters to support even just those languages written in the Latin alphabet. As long as people (and computers) insist on using 8-bit characters, this gives rise to the problem of multiple 'code pages'.

Text encoded according to one code page must be accompanied by separate information stating what code page is to be used. This is difficult in attribute-free file systems such as UNIX and PC DOS, since there is no guaranteed way to keep that information with a text file. Embedding of attribute headers in the file itself is unacceptable.

Few electronic mail systems support the specification of a code page in the message header, although the Internet mail headers are sufficiently extensible that such support could be easily added.

Electronic mail is subject to character set translations, and these are often inconsistent, particularly if the mail has passed through Bitnet nodes or IBM mainframes; multiple code pages increase the likelihood of such corruption.

Switching Between Character Sets

Should it become necessary to switch code pages in the middle of a document (e.g. for a business in Sweden to address a letter to a customer in Turkey), some mechanism must be provided to do so. The ISO 8859 encodings of 8-bit characters define escape sequences that permit changing code pages.

For multi-byte character sets, the situation is more complex. JISCII [7, 8], the Japanese Industrial Standard Code for Information Interchange, is a 14-bit character set defined on a 94×94 grid addressed by two 7-bit characters, using characters in the range 33...126, but biased downward by 32 so that the rows and columns are numbered from 1 to 94. The Chinese GB-2312 and Korean KS C 5601 standards also use a 94×94 grid.

The JISCII character set includes special symbols and punctuation, the printable ISO/ASCII character set, Cyrillic, Greek, the Japanese syllabic alphabets (hiragana and katakana), followed by Level 1 kanji (2965 Chinese characters commonly used in Japanese), and Level 2 kanji (3388 lesser-used Chinese characters). JISCII does *not* include the ISO/ASCII control characters or European alphabetic extensions, nor does the ISO/ASCII subset occupy consecutive positions.

There are at least three ways of encoding documents in JISCII:

- 16-bit characters as 8-bit byte pairs;
- 7-bit ISO/ASCII with shift-in and shift-out escape sequences to enter and leave 16-bit character sections;
- 7-bit ISO/ASCII where character pairs whose first member has the high-order (8^{th}) bit set are taken to be JISCII.

The last two are more compact than the first, but suffer from what may be called the *substring problem.*

Because these two involve a mixture of 8-bit and 16-bit characters, extraction of a valid substring requires examination of surrounding context. In the second method, it may be necessary to scan back to the start to determine whether there is a preceding escape sequence. In the third method, if the first character in the substring does not have its high bit set, one need only examine a single preceding

character to find out whether the first character is a normal one, or the second half of a pair.

Since string searching and substring extraction are among the commonest operations performed on text by a computer, these are very serious drawbacks.

There is also the problem of determining string lengths: is the length the number of characters, or the number of memory cells used to hold the string? Which one is needed depends on the application.

Use of a 16-bit representation eliminates these problems for JISCII, and could as well for a character set that supported all those derived from the Latin alphabet.

However, when Chinese is included, about 50,000 more characters are needed [4]. There are also differences in characters used in the People's Republic of China (due to simplifications instituted after 1949) and those in the Republic of China (Province of Taiwan).

When the 2800 syllabic characters of Korean Hangul are thrown in [4], plus the 900 or so letter variants of classical Arabic [9, 10, 2], and the dozens of writing systems used in India, it seems that even a 16-bit set of up to 65536 characters may be insufficient to cover the world's major languages. Because speakers of Chinese, Indian languages, and Arabic account for more than half of the world's population, these languages cannot be ignored.

Overlapping with the work on ISO 10646 is an effort to develop a comprehensive 16-bit character set called *Unicode*; some Unicode traffic was originally broadcast to the ISO 10646 list, but that practice was discontinued while this article was in preparation. Subscription details are given in the last section below.

The ISO 10646 list review contains the following paragraph:

> As of March, 1990, two coding schemes have emerged. The International Organization for Standardization (ISO) Subcommittee 2, Working Group 2 (SC2/WG2) has developed the ISO 10646 Multi-Octet Code. It is now a "draft proposed" standard (two levels removed from being an international standard). The ISO working group has been working on this project for the last 6 years and it has been subject to unusually wide review for a proposed standard. The other draft standard is the result of the work of a consortium of U.S. companies, mostly from the west coast. It is called Unicode. Both of these draft standards enable the world's communication (newspapers and magazines) and business characters, ideographs, and symbols to be encoded for storage and communication between computers. However, each uses a different approach to making the inevitable tradeoffs.

In my view, Unicode seems short-sighted, and too small. An 18-bit set would probably suffice, so maybe 36-bit machines like our venerable DEC-20, and the UNIVAC 1100 series, will someday be reincarnated! What is more likely, though, is that falling memory prices will make 32-bit characters practicable.

Inadequate Display Support

There is a serious problem of character display. How is a person to read a document that requires characters unavailable on the terminal or printing device? This becomes particularly relevant as we enter an era of international electronic mail and document exchange.

While personal computers and workstations are increasingly offering support for multiple character sets, much remains to be done before the display problem can be eliminated.

Impact on Programming Languages

If character sets are enlarged, computer programming languages must be modified.

TEX 3.0 added only one bit to the character set encoding, and is riddled with assumptions that 256 is the size of the character set. These assumptions are of course introduced in the interests of compactness, so that TEX can run on small machines. With effort, some of these could be eliminated, but probably not all of them; doing so would introduce incompatibilities, and thus lose the right to the name TEX.

With the exception of the ANSI C standard [1], adopted in December 1989, existing programming languages (or at least their compilers) assume 7-bit or 8-bit characters; the last machines using only 6-bit characters were retired in the early 1980s.

ANSI C provides support for 'wide' characters; wide strings take the form `L"..."` and wide character constants are written as `L'..'`. Hexadecimal escape sequences, `\xhhh...` are introduced; they may have any number of hexadecimal digits. The underlying representation of wide character strings may use one or more bytes per character, allowing room for future expansion. Shift-in and shift-out representations are permitted. However, ANSI C states that a byte with all bits zero shall be interpreted as a null character (and therefore, a C string terminator), *independent of the shift state*, and a byte

with all bits zero may not occur in the second or subsequent bytes of a multibyte character. Also, a comment, string literal, or character constant shall begin and end in the initial shift state, and shall consist of a sequence of valid multibyte characters.

Impact on Collating Sequences

Any assignment of characters to a numerical code introduces collating sequence problems.

For example, the Danish and Norwegian alphabets are A...Z, Æ, Ø, Å, while Swedish reverses the order of the last three and uses umlauts: Å, Ö, Ä.

Note that these Scandinavian accented letters are considered *separate* letters; this differs from French and German, which alphabetize such letters without regard to accents. Danish, Norwegian, and Swedish also occasionally use acute accents on the letters 'e' and 'o', for disambiguation of homonyms, and for a few foreign words; these accents are ignored in alphabetization.

With the orthography reform of 1948, Denmark ceased to capitalize nouns, introduced the new letter Å in place of the old Aa, and moved it from the front of the alphabet to the end. Under the reform, Aa is collated as if it were spelled Å, so some people moved from the front of the telephone book to the back. When Aa occurred in proper names, the owners were permitted to retain the old form, so both continue to exist: Aarhus University is in Århus, Denmark, and both the University and city listings are found at the end of the telephone book.

In German, ß ('scharfes s' or 'es-zet') capitalizes to SS (or rarely, SZ), does not occur as an initial letter, and is alphabetized as 'ss'.

In Spanish, 'ch' is treated as a single letter falling between c and d, 'll' is treated as a letter between l and m, and ñ is treated as a letter between n and o.

Although several languages in Eastern Europe and the Soviet Union employ the Cyrillic alphabet, there are variations between countries in both order, and the exact letters used. The reforms introduced after the Russian revolution in 1917 removed some letters from the alphabet, but scholars of pre-1917 literature still require them. A good treatment was given by David Birnbaum in a posting of 30-Nov-1989 to the ISO 8859 list.

The New York Stock Exchange listings are always by corporate abbreviations, yet collation is according to company name; IBM is listed as if it were spelled 'International Business Machines'. Telephone books in some areas move the Macdonalds and the McKays in front of other names beginning with M.

In Japanese and Chinese, the order of ideographic characters is determined by the authors of each dictionary. Many dictionaries base the order on the 214 fundamental 'radicals' (character part building blocks); the dictionary is ordered by groups of characters having the same radical, and within each group, by increasing numbers of strokes, and by pronunciation. However, some characters have more than one radical, many have the same pronunciation (in Japanese, 5500 kanji have only 336 different sounds [6]), and pronunciations may vary with dialects (Chinese has dozens of dialects that are mutually incomprehensible, but share a common writing system). Dictionaries from the People's Republic of China can also be found with ordering according to the Pinyin representation in the Latin alphabet, that is, according to Mandarin pronunciation.

JISCII has yet another assignment of Chinese (kanji) characters into two levels according to frequency of use. In the JIS Level 1 kanji, order is according to dictionary and pronunciation order [11, p. 68]; subgroupings are mostly according to stroke count, with exceptions. In JIS Level 2 kanji, order is according to radical and stroke count. These difficulties have traditionally discouraged the use of indexes in Japanese books, and also seriously impact filing of information in computers and offices.

For a readable account, see the chapter *Practical Consequences of a Large Character Set* in J. Marshall Unger's book [11].

Thus, in many languages, and even in English, sorting according to a collating sequence is a difficult problem, and capitalization cannot easily be changed by a computer program. This has important ramifications for BibTeX, LaTeX, and MakeIndex. In some BibTeX styles, article titles are lowercased, and some LaTeX styles convert titles to uppercase letters; in both, the result is a disaster if the language happens to be German.

Internationalization of Software

The chapters on Native Language Support and Regular Expressions in [12] describe the changes that must be made to the C run-time libraries, and to many UNIX utilities, when extended character sets are used.

For example, international software cannot contain embedded character strings; these must be moved into separate external files that can be customized for each language. In addition, output format strings must be extended syntactically to permit reordering of output tokens (cf. English "The *White* House" and French "La Maison *Blanche*").

Summary and Conclusions

Character coding is a very complex issue, and despite the vigorous discussions on the ISO 8859 list, I do not see a solution on the horizon. Because it uses a different character set (EBCDIC) than everyone else, IBM will be affected more by character set issues than other vendors; its conservatism, and historical slowness to respond to the demands of the market and its users, also suggests that solutions will not soon be forthcoming.

In my view, the advent of support for 8-bit characters in TEX 3.0 will for some time *hinder*, rather than help, document portability. There is a conflict between the desire for ease of use and readability of the input file on the part of the author or typist who enters it by, say, striking the Ø key on a Danish keyboard, and the co-author in Britain who cannot display the Ø on the screen, and may have no idea what character was intended.

Authors who stick to the 7-bit ISO/ASCII character set and with some labor, enter `\O{}` instead of using the Ø key, will promote document portability.

Alternatively, translation filters will be needed, but it may not be possible to base them entirely on simple text substitutions, at least in the 7-bit to 8-bit direction, since `\O` cannot be substituted if it is the initial part of another control sequence. Also, in TEX 3.0, hyphenation opportunities will be lost if accented characters encoded as single 8-bit values are replaced by control sequences.

BIBTEX, LATEX, and MakeIndex will require revisions in the future for

- support of 8-bit character sets,
- more flexible provision for specification of sorting order,
- suppression of capitalization changes.

Even TEX itself may need modifications, since the *xchr*[] character translation array is initialized early in the program to values which depend upon the local character set, and no provision is made for switching code pages dynamically.

Joining the Mailing Lists

To subscribe to the ISO 8859 or ISO 10646 mailing lists, send an e-mail message to the server

```
LISTSERV@JHUVM.BITNET
```

with the *body* text (`LISTSERV` ignores the e-mail *Subject:* line)

```
SUBSCRIBE ISO8859 <your-personal-name>
```

or

```
SUBSCRIBE ISO10646 <your-personal-name>
```

Letter case is ignored in `LISTSERV` commands.

Your e-mail return address is automatically extracted from your mail message. The personal name is used to annotate the mailing list, which can be retrieved with a message like `REVIEW ISO10646`, in case you would like to know your correspondents by other than cryptic e-mail addresses. The `REVIEW` command also provides a summary of the purpose of the discussions.

All list traffic is archived; a message with the text `INDEX ISO10646` will retrieve an index for that list, and a following message with the text `GET ISO10646 filetype` will fetch a particular file. For more details on `LISTSERV`, send a message with the text `INFO GENINTRO`.

To get on the Unicode list, send a message requesting inclusion to Glenn Wright:

```
glennw@sun.com
```

References

[1] American National Standards Institute, 1430 Broadway, New York, N. Y., 10018. *American National Standard Programming Language C, ANSI X3-159.1989*, December 14 1989.

[2] Joseph D. Becker. Arabic Word Processing. *Communications of the Association for Computing Machinery*, 30(7):600–610, July 1987.

[3] Janusz S. Bień. On Standards for Computer Modern Font Extensions. *TUGboat*, 11(2):175–183, June 1990.

[4] S. Duncan, T. Mukaii, and S. Kuno. A Computer Graphics System for Non-Alphabetic Orthographies. *Computer Studies in the Humanities*, 2(3):113–132, October 1969.

[5] Yannis Haralambous. TEX and Latin Alphabet Languages. *TUGboat*, 10(3):342–345, November 1989.

[6] A. V. Hershey. Calligraphy for Computers. Technical Report TR-2101, U. S. Naval Weapons Laboratory, Dahlgren, Virginia 22448, August 1967.

[7] Japanese Standards Association, 1-24, Akasaka 4 Chome, Minato-ku, Tokyo, 107 Japan. *Japanese Industrial Standard JIS C 6626-1978 Code of the Japanese Graphics Character Set for Information Interchange*, 1978.

[8] Japanese Standards Association, 1-24, Akasaka 4 Chome, Minato-ku, Tokyo, 107 Japan. *Japanese Industrial Standard JIS C 6234-1983 24-dots Matrix Character Patterns for Dot Printers*, 1983.

[9] G.A. Kubba. The Impact of Computers on Arabic Writing, Character Processing, and Teaching. *Information Processing*, 80:961–965, 1980.

[10] Pierre Mackay. Typesetting Problem Scripts. *Byte*, 11(2):201–218, February 1986.

[11] J. Marshall Unger. *The Fifth Generation Fallacy—Why Japan is Betting its Future on Artificial Intelligence.* Oxford University Press, 1987.

[12] X/Open Company, Ltd. *X/Open Portability Guide, Supplementary Definitions*, volume 3. Prentice-Hall, 1989.

◇ Nelson H.F. Beebe
Center for Scientific Computing
and Department of
Mathematics
South Physics Building
University of Utah
Salt Lake City, UT 84112
USA
Tel: (801) 581-5254
Internet: Beebe@science.utah.edu

On Standards for Computer Modern Font Extensions

Janusz S. Bień

Abstract

Haralambous' proposal to standardize the unused part of Computer Modern fonts is discussed, and some modifications and extensions suggested. The idea is pursued by designing the extended CM font layout, and an example is given for one of its possible uses.

1 Introduction

In my note [4] I advocated an old ([15, p. 46], [6, p. 45]) but rarely used idea to place national letters (actually, the Polish ones, but the generalization is obvious) in the unused part of Computer Modern fonts, i.e. as the characters with the codes higher than 127; this approach allows the handling of national languages in a way upward compatible with the standard (American) English TEX. A similar proposal was made independently by Yannis Haralambous [8], who states also that the use of non-English letters of latin alphabets should be coordinated, resulting in a single widely used extension to Computer Modern fonts — I strongly support the principal idea, and I pursue it in the present paper. To organize the discussion in a systematic way, I will use the notions — borrowed from [2] — of text *encoding*, *typing* and *rendering*.

2 Text encoding

In the context of TEX, *encoding* means the character sets of the fonts in question and their layouts. In the present section I will focus my attention on the character sets, as the layouts should be influenced, among others, by *typing* considerations.

In an attempt to obtain a general idea about the use of the latin alphabet worldwide, I looked up the only relevant reference work I am aware of, namely *Languages Identification Guide* [7] (hereafter *LIG*). Apart from the latin scripts used in the Soviet Union and later replaced by Cyrillic ones, it lists 82 languages using the latin alphabet with additional letters (I preserve the original spelling):

> Albanian, Aymara, Basque, Breton, Bui, Catalan, Choctaw, Chuana, Cree, Czech, Danish, Delaware, Dutch, Eskimo, Esperanto, Estonian, Ewe, Faroese (also spelled Faroeish), Fiji, Finnish, French, Frisian, Fulbe, German, Guarani, Hausa, Hungarian, Icelandic, Irish, Italian, Javanese, Juang, Kasubian, Kurdish, Lahu, Lahuli, Latin, Lettish, Lingala, Lithuanian, Lisu, Luba, Madura, Miao, Malagash, Malay, Mandingo, Minankabaw, Mohawk, Mossi, Navaho, Norwegian, Occidental, Ojibway (also spelled Ojibwe), Polish, Portuguese, Quechua, Rhaeto-Romanic (Ladin, Romansh), Rumanian, Samoan, Seneca, Serbo-Croatian, Sioux, Slovak, Slovene, Spanish, Suto, Sundanese, Swahili, Swedish, Tagalog, Turkish, Uolio, Vietnamese, Volapük, Welsh, Wolof, Y, Yoruba, Zulu.

This list includes some languages and dialects with no script at all, for which the information supplied concerns more or less standard transcription. For most of them this fact is noted explicitly, but the exception of Kasubian (usually recognized as a dialect of Polish) suggests that this is not always the case. I noticed some inconsistencies in the numerous indexes to the book, but only one omission (described later) in the proper text. Of course, it is difficult for me to judge the reliability of the work as a whole.

The number of additional letters in the latin alphabets listed in *LIG* — including some variants of shape but excluding upper case letters — is 176.