The Significance of Unicode

Yu Sung
Advisor, National Central Library
Taipei, Taiwan, R.O.C.

I. Introduction

The Pacific Ocean is rimmed by vast lands of various cultural heritages. Among the peoples and cultures of Pacific neighborhood, there are constant interactions and interrelations in many aspects – political, commerce, life style, religion, philosophical, knowledge etc. Therefore a modern person, be it a man on the street, or a scholar, or a businessman, has a legitimate need to access knowledge and communicate across borders. However, the Pacific Rim is also full of linguistic diversity. Although in the age of computer and telecommunication, there have been constant development efforts to make communication easier, faster, more economical, and less cumbersome. But the fact remains that we are still far from the vision in which information can be freely and easily exchanged regardless of the language or script in which the data is recorded. There are many causes of this situation. A most fundamental one is multiplicity of conflicting and cumbersome standards. Unicode is a new character code standard for computer and telecommunication systems encompassing nearly all major languages, plus quite a few minor ones, in the world, in order to improve the situation. This paper describes the various codes prior to Unicode and discusses the concept, design and implications of this code. Also its shortcomings and problems to be worked out.

II. Character Codes: A Brief Review

Although the computer was invented to compute mathematical problems, It was soon adapted to handle text. Characters are basic units of text. For example, English letters of upper and lower cases, numeral digits, and punctuation and special marks are all characters. Encoding is the method of assigning numeric values to characters to represent the latter for computer text processing. The basic requirements for an encoding system are it must be comprehensive, unambiguous, easy to use and it should have good support. Comprehensive means the coverage of the character set is adequate for the applications intended. With English and other alphabetical languages this requirement is not a problem, because these alphabets are finite sets of several ten letters. However, for Far Eastern languages this requirement is not that easy because the character sets are huge, open-ended and size of the

character set depends on applications. The requirement of being unambiguous means there should be a one-to-one relationship between characters and assigned codes. No character should have more than one assigned code. (Note: this rule sometimes is waived in special situations.) Also vice versa – no code value should correspond to more than one character. Easy to use means easy to input and edit. For alphabetical languages this again presents no great problem. As to East Asian languages it is far more difficult because the size of the character set is in the order of ten thousands to hundred thousands, and each language has complicated inputting rules, which require special inputting editors. Support is important because every system needs constant improvement and bug elimination. Also in a character code, in addition to what we call graphic characters, a few special characters for various control purposes were included.

When we discuss development of high technology, cost and state of technology determine the architecture. So in the early days of computers both the primary and secondary memories were expensive, it was important to reduce the data size. In the beginning character codes of 5 and 6 bits (e.g. Baudot and BCD codes) were used, which were very inconvenient according to today's standards. For instance, only letters of upper case are represented. Despite the handicaps, people managed to use it in accounting, telecommunication, and even library automation applications.

By 1963 the demand for better representation succeeded in producing what is now known as ASCII (American Standard Character Code for Information Interchange), a 7-bit English-language code. It included the characters mentioned in the beginning of this section – 52 English letters of upper and lower cases, 10 digital numerals, 32 punctuation/special marks plus 34 control characters. The 94 printable characters sometimes are called graphic characters to distinguish them from the 34 control characters. This code table has been used for 37 years now, and generally speaking, it is satisfactory for its original purpose of representing English text. Occasionally maybe someone would wish for more punctuation/special marks.

However, when text processing is moved to other countries in Europe, ASCII has to be either modified or extended to accommodate diacritics and more graphic characters. ISO-646 allows substitution of "national use characters". There have been various variant code pages, most of which have different code assignments. Also there were special purpose code pages such as ANSEL. The resulted confusion is not hard to understand. Whenever one tries to send data to a different computer, one cannot guarantee the two computers use the same code table and understand each other. It is a problem of conflicting standards, or to put it in another way, proliferation of code pages. Eventually a series of

standard code pages (ISO 8859-series) came out, each for one locale. These code pages share a common ASCII part, works a little better, but do not eliminate the problem.

As for East Asia the problems are even greater. Though many countries use Chinese or Han characters in their written languages, there are significant differences in the character sets, usage, and glyphs. The sizes of the character sets range from several thousands to tens of thousands, depending on application and country. So the East Asian character codes are usually two bytes, some three or even four bytes. Also they need to mix with 1-byte ASCII and other codes. Therefore the processing of these texts with such codes is slow and complicated. Unfortunately there is also multiplicity of standards, and it is difficult to communicate. As there are different code pages between different countries, one needs to switch code pages, which is slow and prone to errors. There is no intermixing of different languages.

However, despite the difficulties and confusions, computer text processing has become more widely used. And the stage is also set for a new start.

III. The Unicode Solution

As early as 1989, ISO formed a working group to study the architecture of a universal character set and code to consolidate major languages of the world. The original plan was to combine all national code pages into a 4-byte code. Each country has its own code section. This plan met some resistance because computer vendors thought 4-byte character was excessive and the proposed structure was inefficient and cumbersome. So the draft proposal was defeated in ballot. Before this happened, a group of vendors and organizations such as Apple, IBM, Microsoft, RLG, NeXT, Sun Microsystems and Xerox and others who rebelled against the original proposal had founded the Unicode Consortium to develop an alternate plan. Then ISO adopted the Unicode plan and from that date on Unicode and ISO-10646 (or UCS) for all practical purposes are the same.

Unicode is a 2/4-byte or 16/32-bit code, with characters of the most widely used languages assigned in the 2-byte plane called BMP (Basic Multilingual Plane). For Unicode Standard 3.0 soon to be released there are 10,236 alphabetics and symbols, 27,786 CJKV ideographs, 11,172 Hangul syllables, totaling 49,194 characters assigned in BMP. As to the characters in other planes, Unicode now uses two special 2-byte surrogates to designate them. The assignments of the 4-byte characters are not released yet, but it is known it is in process. The present plan allows 16 additional planes (Plane 1 through 16) beyond the 65,536 of the BMP (Plane 0), totaling 1,114,112 code points. At least for now

this number is staggering to us working on character encoding. I presume it will be a while before all these code points used up. There are 6,400 code points in BMP and 131,072 in the 4-byte planes reserved for users' private use.

Unicode's important features are described in the following.

- To represent characters, Unicode uses fixed 16-bit codes in BMP. For characters beyond BMP, it uses combination of two fixed 16-bit surrogate codes. The surrogate codes are in specially reserved blocks, so do not conflict with regular 2-byte Unicode character codes. There is no 8-bit encoding, so parsing is much easier.
- Since Unicode treats 16 bits as a unit, the control codes are also 16 bits. There are altogether 64 control codes (0000-001F, 007F-009F) in the whole 65,536 code points. This is different from the other 2-byte codes, in which the first 32 positions of both bytes must not be used for character encoding, resulting in much reduced usable coding space.
- Unicode encodes characters, not glyphs. Each character has a unique code and name with no ambiguity. However, there may be a one-to-many or many-to-one relationship between characters and glyphs.
- Unicode tries to unify languages whenever possible. Thus all western European languages are grouped under Latin, all Slavic under Cyrillic etc. Chinese (Mainland and Taiwan), Japanese, Korean, and Vietnam languages are unified under UniHan, saving quite a lot coding space. For instance, there are 27,484 Han characters in BMP. If counted separately, the number would be at least doubled.
- Unicode is a consolidation of all language codes in one big table, therefore does not need ISO-2022 type escape sequences to switch code pages. So it is easy to intermix different language characters in display or printed pages.
- The target of Unicode is 'universal', to include all the languages in the world. Now it includes over 30 languages and has about 50 in the process pipeline.
- BMP contains all the major alphabetic languages and the more common characters of UniHan. The less frequent used languages and UniHan characters are put in 4-byte planes. This arrangement admittedly is a compromise, but it is believed to be an acceptable one.
- Unicode does not include commercial logos or fantasy scripts like Klingon.
- Besides the common left-to-right directionality, Unicode also allows right-to-left languages such as Arabic and Hebrew. The vertical top-to-bottom directionality is under study.

From the known features above, we can have a picture of how Unicode operate. First, Unicode is a big, universal code table which now includes all major languages and quite a few minor languages. It will be expanding to include more languages used by communities of five million or more in time. Also all scripts of any historical importance will be included. Although there is still space in BMP now, it will be less and less available. The criteria to be in BMP probably will be on the usage of candidate languages. Nevertheless the 4-byte planes have quite a lot of "real estate". If a language gets assigned there, it will be slower in processing and take up more memory when stored. But this is a small price to pay to be in the universal system.

Second. Unicode is only a code table. One needs input method editors and font files in order to input and output characters. These are language dependent, and are under development by vendors. As far as we know, there are no serious problems here. Maybe it is a matter of more time and more investment. Since it is one table, it does not pose special problems in intermixing characters from different languages.

Third. Compared with the previous method of treating ideographic languages, which mixes 1- and 2-byte characters in strings, parsing and processing is much straightforward, and thus faster and less error-prone.

Fourth. By allowing right-to-left directionality and in the future vertical directionality, Unicode aims at becoming a truly universal character code system.

Fifth. For data communication many protocols impose constraints on allowable text characters. So Unicode uses transformation formats to convert the text to enable it for transmission on networks. There are three transformation methods in present use – UTF-7, UTF-8 and UTF-16.

Sixth. At present Unicode-based software is comparatively few. If some software has an international market, it is much more advantageous to build it on Unicode because it saves a lot of time and money in internationalization and localization, and generate more sales. One example is Microsoft Office 2000, which according to Microsoft, could generate additional $300 million profit per year because of Unicode.

Seventh. The shortcomings of Unicode are the facts that it takes more space in primary and secondary memories than ASCII and other 1-byte codes, and that the processing system will be bigger. Therefore for certain applications whose usage is limited to a region a non-Unicode solution still has merits. Also as Unicode does not define glyphs

and font design is in the hands of font designing vendors, it is possible occasionally there are errors in glyphs. However, these occurrences should be rare and correctable. More likely to happen is one does not like the style or flavor of the glyph.

Eighth. Another inherent weakness of Unicode is its hierarchy of decision making which makes any request for changes a long and slow process. For East Asian characters there are times when characters should be added or erroneous ones should be corrected. So some makeshift procedure should be devised to allow temporary repairs in users' private area by some user group. The regular hierarchy then can do their review and deliberation on the case, to rectify or disapprove.

IV. Conclusion

A standard in itself does not determine a technology's success or outcome. However, a standard that answers a collective need and provides a feasible way to satisfy the need is a significant endeavor. WWW is such a standard. I believe Unicode is another. It will make the seeking and retrieving international information on the web and exchanging multilingual computerized data on Internet a much easier thing than before.

References

The Unicode Consortium. The Unicode Standard, Version 2.0. published by Addison-Wesley Developer Press. 1996.

Jukka Korpela. A Tutorial on Character Issues. http://www.hut.fi/u/jkorpela/chars.html.

Jack Cain. Linguistic Diversity, Computers and Unicode. Presented at Networking the Pacific: An International Forum. May 5-6, 1995. Victoria, B.C. Canada.

Torry Graham. 10646 and All that: Unicode, ISO 10646, and the Quest for a Universal Character Set. http://www.mulberrytech.com/papers/unicode/sld001.htm.