Features of M Required in China: Insights from Translating FileMan

by Feng Huang

Abstract

This article describes a complete Chinese translation of the VA File Manager. This process required careful analysis of features of M required for supporting Chinese language applications as well as some prerequisite considerations relating to ways that the Chinese language can be adapted to support M.

Introduction

The VA FileMan is a database-management system package with fourth generation features. Written in M, FileMan was designed to provide an easy way for end users to create and manipulate databases of many types. There are, however, some problems inherent in attempting to adapt this package for use with the Chinese language. To begin with, health professionals in China are usually unfamiliar with English, so that translating the user interface is an essential requirement for its use in China.

Writing in Chinese is fundamentally different from writing in languages using the Latin alphabet. The differences are numerous, but the principal conclusion is that FileMan must be modified extensively for it to serve Chinese-speaking users.

Together with Professor Zheng Te, I finished the first translation of VA FileMan (version 17.07) in late 1988. Beginning in 1989, we used this version to build an administrative medical-records management system for the master files in our hospital. Since then, we have completed translation of VA FileMan versions 18.0 and then 19.0. Both of these versions have been used in various applications in our hospital.

Computerizing the Chinese Character Set

The Chinese character set differs substantially from the concepts embodied in an alphabetic language. Many of these concepts are covered in other documents and will not be treated in detail in this article.[1,2] However, a few points are pertinent to the FileMan translation problem discussed here. Chinese use nonalphabetic symbols, often referred to by linguists as logograms, although the term ideogram is sometimes used in informal discussion. We use the term *character* to refer to one such logogram, which may represent a complete or partial *word* in English equivalency in a Chinese sentence. It is important also to realize that the Chinese language does not use spaces to separate words or concepts; all characters are listed without breaks (with the exception of occasional midsentence punctuation marks) until the final sentence terminator.

In general terms, English and Chinese may be characterized in the following syntactical description:

Example 1: English

```
<sentence>  ::=<word_unit>...
<word _unit>::=<word><punct> | <word_unit>
     <word> ::=<alpha>...
     <punc> ::=Tp (punctuation characters including
               space)
     <alpha>::=Ta (any of 52 letters: A-Z, a-z)
```

By contrast:

Example 2: Chinese

```
  <sentence> ::=<char_unit>...
  <char_unit>::=<character>[punct] | char_unit>
       <pun> ::=Tp(punctuation characters including
                 space)
  <character>::=Tk(more than 50,000 Chinese
                 logograms)
```

These conceptual differences must be taken into account when translating text for a program such as FileMan into Chinese.

From this description, it is clear that the Chinese character (or logogram or glyph) often serves more the role of a word than the equivalent of an English letter. For example, *Who are you?* in Chinese is written <ni><shi><shui> (transliterated using the Pinyin phoneticization of the Chinese characters). The English equivalent consists of three words separated by spaces, whereas in Chinese there are three Chinese characters with no spaces. In this particular interrogative sentence, each Chinese character corresponds to a word in English:

Who	-- <shui>
are	-- <shi>
you	-- <ni>

In other words, a Chinese character may serve the role of an English character, a phoneme, or a word.

In the 1980s, the Chinese National Standards Institute published the character set GB 2312-80, which was given the name "Code of Chinese logograms set for information interchange—the primary set." This standard character set includes 6,763 Chinese characters, which together encompass more than 99.99 percent of applications. This standard has the following features:

- Each Chinese character is defined by a 2-byte (14-bit) code.
- The codes used for each byte of a Chinese character range from 33-126. For this reason, the characters do not conflict with ASCII control characters, and retain compatibility with GB 1988-80 (ASCII).
- The codes are arranged in a matrix of 94 x 94 positions, with a possible 8,836 different codes available (of which 6,763 are defined as Chinese logograms). The values of the codes can be represented by the 94 graphic characters of the ASCII character set, beginning with `!` for position 1, and ending with `~` for position 94 in each row/column (following the ASCII code values beginning at 33 (`!`). The code values for 1-32 are not used so as to avoid any possible confusion with ASCII control characters. This approach makes it possible to identify code positions by two graphic ASCII characters, one representing the row, the second the column. For instance, the sentence <ni><shi><shui> can be expressed as Dc,JG,k- (or 36-67,42-39,75-13). Using this coding scheme unifies the character and word concept for the Chinese language.

Another way of looking at the representation of Chinese words is to use 7-bit characters preceded by some unique control code, such as the "Shift-In" ASCII control character (15). This code would signify that the next two ASCII characters identified the row and column of a Chinese character. Using this method, the string <SI>Dc<SI>JG<SI>k- would define the same three characters. This technique is still used in some network systems in China.

However, the most widely used form of coding Chinese characters involves the 2-byte code in which the high-order (8th) bit is always set to 1 so as to avoid conflict with ASCII codes, and retaining the remaining 7-bit codes of each byte for the row/column positions, respectively. Using this form, the syntax for Chinese characters can be expressed as:

Example 3: Chinese

```
<sentence> ::=<word_unit>
<word_unit>::=<word>[punct] |<word_unit>
     <word>::=<R-octet><C-octet>
     <punc>::=Tp(punctuation characters including
                space)
  <R-octet>::=Tl(any of the 8-bit codes 161-254, the
                8-bit equivalents of ASCII 33-126)
  <C-octet>::=Tl(any of the 8-bit codes 161-254, the
                8-bit equivalents of ASCII 33-126))
```

where *R-octet* refers to the first byte and *C-octet* refers to the second byte of a 2-byte Chinese character. The letters *R* and *C* refer to the definition of *row* and *column* position of each character in the 94 x 94 matrix described earlier. The R- and C-octets are sometimes referred to as the *physical* bytes required to represent a *logical* Chinese character. This syntax is similar to Example 1. The three syntax descriptions can be compared as follows:

```
1.  7-bit ASCII set: Ta TP TNC -- -- --
2.  8-bit ASCII set: Ta Tp Tnc To Tl --
3.           G1 set: Ta Tp --- -- -- --
4.           G2 set: -- Tp --- -- -- Tk
5.           G3 set: -- Tp --- -- Tl --
```

where `To` is any of the 8-bit control characters 128-160 and `Tnc` includes numeric, the 33 control characters including 128 (Del).

It is evident that both the English (Example 1) and Chinese (Example 3) sets can be supported by 8-bit ASCII codes, which explains why many Western packages that support 8-bit ASCII can be used to process Chinese characters without modification.

The distinction between physical characters (bytes) and their use to represent logical entities (e.g., English or Chinese characters) is unimportant in the use of the Latin alphabet, but it affects users of M in the use of functions such as `$ASCII`, `$C` and the string functions `$LENGTH`, `$EXTRACT`, etc. It is necessary to establish whether, for instance, `$LENGTH` refers to the physical number of bytes or the logical number of characters in a string of codes. M implementors, faced with multiple-octet character sets, took different paths in their treatment of such characters, as described in the next section.

Chinese Characters in M

M was originally defined for use with the English language using ASCII characters. As the power of M became more widely accepted, however, it was quickly exported to countries in which English was not the primary language. These languages required the use of expanded character sets, which in turn required the use of non-ASCII characters. In the first years following standardization of M, many nonstandard implementations of the language were introduced in countries besides the United States and United Kingdom, notably Europe, Brazil, and Japan. Initially, vendors were able to use 8-bit character representations to handle expanded alphabets (including the Japanese phonetic Katakana character set). Each of these implementations, however, was nonstandard and in most cases incompatible with other national implementations. In time, it became necessary in countries with logographic characters such as Chinese and Japanese to accommodate 2-byte characters of the type described earlier in this paper. Once again, nonstandard methods were used, some of which are described briefly in this article.

In recent years, the MUMPS Development Committee (MDC) has faced this issue, and the newly proposed standard for M has made it possible to permit non-ASCII character sets to be accommodated by a standard version of M. These new features, described in detail elsewhere, are most welcome in countries such as China; they will eventually lead to new standard implementations of M that will be portable across hardware and operating system platforms.[3]

When we began the process of translating FileMan into Chinese, however, there was no uniform method for treating non-ASCII characters. In some implementations, the character was treated as a physical character (one byte representing a character, regardless of its true meaning in ASCII or the code described in Example 3 above). Other implementations treated Chinese characters as 2-byte pairs which together were considered a single, logical character. The problems we faced, therefore, while historic in nature owing to the new standard, represent an interesting problem of translation to accommodate nonstandard versions of M.

We have attempted to implement Chinese FileMan using both types of implementations. Because the 2-byte logical character approach is more consistent with the evolving multilingual M standard, we present expanded descriptions of the logical character, and only allude briefly to the physical-character method. Nevertheless, some of the problems encountered will require treatment when a Character Set Profile for GB 2312-80 is adopted.

Several features must exist in an implementation of M that deals with Chinese characters. They include: character set; pattern match code; name; interpretation of GB 2312-80 characters; and collation.

Character Set

Logical Chinese characters require the use of two bytes for each logical character. If 8-bit characters are used, the codes 161-254 can unambiguously represent their 7-bit ASCII equivalents 33-126 without being confused with the ASCII characters. In this representation, no distinction is required to interpret the single byte ASCII codes from the two-character representation of the Chinese characters. Using 7-bit codes for all characters is possible, but that creates a number of difficulties, in that Chinese operating systems currently depend on 8-bit codes, and patcode recognition is much more difficult.

Pattern-Match Code Option

The pattern-match code option must accommodate Chinese characters. The pattern-match process is one of the strongest features of M. Although it might be possible to extend existing definitions of the ASCII patcodes (e.g., by assigning all Chinese logograms to the patcode E), the most logical approach seems to require use of one or more new codes to uniquely identify Chinese characters. The problem is made more complex by GB 2312-80, which was patterned after the Japanese Industry Standard, JIS C-6226. GB 2312-80 includes not only the 6,763 Chinese logograms but also separate code areas for the Latin, Cyrillic, Greek, Japanese Katakana and Hiragana phonetic characters, plus an extended set of punctuation characters including those found in ASCII but also including others not found in Western punctuation character sets. The multiplicity of different phonetic characters will require special treatment, but these problems lie in the future, and we did not attempt to resolve them. Instead, we chose to use a (temporarily) nonstandard patcode K to represent characters written using the GB 2312-80 code set, and to allow continued use of the ASCII patcodes, retaining their current meaning.

While we recognize that this is an imperfect solution, it permitted us to complete translation of FileMan without having to resolve the more complex issues just described. To further simplify the process, we restricted our use of GB 2312-80 to representations of Chinese only, retaining English usage in FileMan where appropriate and using the ASCII patcodes to clarify those portions of the FileMan code.

Name

The definition of codes permissible for naming variables in M should be extended so that Chinese characters can be used. Fortunately, the use of indirection made this task somewhat simpler in our FileMan translation process. But this issue, which has been resolved in the proposed new standard, poses some problems in earlier implementations.

Interpretation of GB 2312-80 Characters

Since all GB 2312-80 characters are two bytes long, implementations of M must recognize these 2-byte units as single characters to avoid any possible division of bytes on other than along character boundaries. Two additional complications are presented in Chinese because there are no spaces between words, and words may consist of one, two, or even more Chinese logograms. Clearly, the use of string functions must avoid splitting characters, but it is also considerably more difficult to separate words, since the correct separation requires considerable knowledge of the language rather than the simple use of the space character.

The internal codes used to represent mixed strings of ASCII and GB 2312-80 have taken various forms in different M implementations. These differences include the use of escape sequences to provide separations between different code sets, the use of uniform 2-byte codes for both ASCII and GB 2312-80 (with the 8-bit used to distinguish an ASCII character whose second byte is ignored), and even in early implementations' use of an escape sequence to identify each non-ASCII character. As a result, $LENGTH requires more sophisticated processing. This feature should be transparent to the user, since the convention adopted by any single implementation is consistent in that version.

Collation

Collating Chinese characters represents a much more difficult problem than in English. The new proposed standard provides for an algorithmic approach to collation that will become a part of the Character Set Profile, and it is certain that the Chinese will need to adopt some algorithm for its character set. This problem becomes somewhat more acute because GB 2312-80 divides its Chinese logograms into two sets or levels. The first set is ordered according to the Pinyin pronunciation, whereas the second utilizes the conventional "radical and stroke count" sequence, which is unrelated to the pronunciation of the character. These problems are alluded to in the specific problems we encountered in translating FileMan, but a more comprehensive solution will require additional study.

The language features just described represent an overview of some of the more serious problems associated with manipulation in M of Chinese language elements. While our solution was temporary, pending adoption of a new Character Set Profile for the Chinese language, the experience is helpful to illustrate the need to resolve these issues in the near future, so that standard Chinese implementations of M can be used throughout our country.

Translation of the VA FileMan for Chinese Users

Translation of VA FileMan consisted of three main tasks: modifying the pattern match operator in M; adapting the user-interface messages for Chinese text; and allowing for special cases.

DOUBLE YOUR PRODUCTIVITY!

Our **MEdit™** full-screen routine editor and customizable **MShell™** toolkit will cut your development time, and make multi-platform development a snap!
We also offer expert consulting services for system management, custom software, health care, and much more!

Call 1-800-370-1935

McIntyre Consulting, Inc.
336 Baker Ave., Concord, MA 01742
(508) 371-1935 Fax: (508) 369-6693
Email: msm@mcinc.com

A Few Good People

The Information Systems department of Kennedy Memorial Hospitals - University Medical Center is experiencing unprecedented growth, with new development efforts and the launching of information dependent organizational initiatives. We are looking for a few good people to join our in-house, non-vendor application systems group to participate in and contribute to this growth. The following positions are available:

Project Leader
Senior Programmer/Analyst
Programmer/Analyst

These positions are responsible for development, support, and maintenance of hospital information systems, including the investigation and integration of new technologies. The successful candidates must posses excellent technical, analytical, and communication skills, along with hands-on hospital application experience.

We offer an environment which promotes technological advancement and career growth in the expanding field of healthcare. If you are interested in pursuing the challenges which exist at Kennedy, please forward your resume, with salary requirements to:

Kennedy Memorial Hospitals
MIS Department
500 Marlboro Avenue
Cherry Hill, NJ 08034-5084

an equal opportunity employer

Modifying the Pattern-Match Operator

This task was undertaken to serve two goals: first, to permit FileMan to manipulate Chinese characters; and then to adapt the code to perform pattern checks on Chinese characters as is done in the English version.

FileMan contains many data entry checks that rely on pattern-match verification. If the implementation has not been extended to handle Chinese logograms as a separate pattern (while retaining the existing Latin Alphabetic patcodes), then the pattern check must be relaxed from `X?.ANP` either to `X?.E` or `X?.KUNP`. If the patcodes are modified in the more comprehensive form described earlier, few changes are required for the existing code.

Further, if the definition of a name in M is not extended to include ideographic characters, then expressions such as `X?1"^"1A.A1"("` `(DICATT5)` and other comparable pattern expressions need to be adapted to accommodate extended name conventions.

Adapting the User Interface Messages for Chinese Text

This aspect of the translation consisted of a relatively straightforward translation of user messages and screen formats to displays that would seem natural to Chinese users, requiring slight adjustment of the text output format to accommodate these changes.

Allowing for Special Cases

A few special cases had to be taken into consideration in the translation process. First was the retention of certain English responses. At first, we attempted to replace all English text to be entered by the user with Chinese equivalents. We soon found, however, that the Chinese translation was more cumbersome. Instead we returned to the English original. Built-in abbreviations, such as *Y* for Yes and *T* for today's date, proved much easier to enter than Chinese equivalents. For this reason, we retained most commands, functions, and abbreviation names in their original English form. Chinese users need to remember only a few English words, and in so doing they greatly simplify the interactive process of specifying keys for searching. Although the Chinese version permits use of approximately sixty built-in functions, Chinese equivalents were defined and can also be used if preferred.

Second was changing the date format. In FileMan, the date is stored in internal format as, for example, "2920101" to represent January 1, 1991 (the external format frequently used in FileMan). Chinese users are confused by the English names of months, so the visual output of the date must be modified. This transformation is required both for input of dates by users and for their display. (The internal format remains unchanged.)

For output, the following changes are made. The English version of FileMan uses a convention of *M D Y* signifying month, day, and year respectively. The conventional format for date in Chinese is Y <nian> M <yue> D <ri>, using the Chinese equivalent words for each component of the date. Chinese users can accept either Y/M/D or M/D/Y formats, provided that the values are numeric (instead of the English names for each month). If the routine to be modified simply displays the date, then these changes are simple, requiring only modification of a few FileMan routines (e.g., `DIO2`, `DIO3`, `DIPZ2`, `DIQ` and `DINIT3`).

In the case of input, a date entered by a user is checked by the routine `%DT`. If the date is entered in the Chinese format (Y/M/D), this routine will not perform a correct error check on user input. Since Chinese can accept the M/D/Y format, it is sufficient to modify the code of `%DT` to replace the month abbreviations (Jan, Feb, etc.) with numeric equivalents (1, 2, . . .). Using this approach, the user is asked to type in a date as M/D/Y using numeric values, and `%DT` evaluates it correctly.

A third special case is word arrangement. As noted earlier, Chinese text does not use spaces between words. Since the internal code for each Chinese logogram consists of two ASCII graphics, confusion may exist when Chinese text is mixed with true ASCII characters. This problem affects FileMan in two ways.

Fourth is keywords. Keyword references are important for lookup functions. In English, keywords are readily identified by separating characters such as spaces or other delimiters. Separators of this type are not normally used in Chinese. When keywords are required for searching in Chinese, some form of delimiter must be introduced when the keyword is input by the user. Two methods can be used to obviate this input requirement. One method is selection from a choice of predefined keywords. This approach stores a list of key phrases and allows the system to reference these choices automatically. Another method is using frequency-determined keywords. Since many Chinese words (phrases) consist of compounds of two Chinese characters, it may be possible to assign keyword status to the first of two such characters if the frequency of occurrence of that phrase exceeds some defined threshold.

The final problem is wrap-around text. In English, where word boundaries are important, it is necessary to provide a wrap-around algorithm that deletes a partial word at the end of a line in order to reproduce the entire word on the next line. This function is not needed in Chinese text, where word wrapping is not required. The solution to the Chinese version of FileMan is to treat English and Chinese text differently. A routine, `DIWP`, is used as follows:

- If a text is longer than one line, combine the two lines into a single line L.
- If the line does not contain a Chinese character (if `L'?.E1K.E`), return to normal English word-wrap logic.
- Or else, if the last character is not a Chinese character, `($E(L,width))?1K` truncate the line at that point and return.
- If the line ends with a Chinese character, set CNT=width and inspect each preceding character until a non-Chinese character is found, and break the text at that position.
- If the physical length is an even number of bytes (i.e., the line ends on a Chinese character boundary), use the existing line and return.
- Or else, remove one more byte and use that length for the line.

(Note that this algorithm assumes that physical byte counts are used in functions, rather than logical characters.)

This algorithm separates a string of Chinese characters without the possibility of splitting a character. The same algorithm is included in `HEAD^DILO` to generate a vertical title of Chinese Characters. Note, however, that some problems associated with different widths of Chinese and ASCII characters may further complicate the display format. We bypassed this issue by assuming the wider size for each character.

Soundex

Soundex is used in Western countries to search for names. In Chinese, this principle is also important. The Soundex algorithm works well for looking up Western names. But in its Latin form, Soundex does not work well with Chinese names, which are often monosyllabic, generating the same code for a large number of names. The same principle can be applied to Chinese characters. In GB 2312-80, there are 6,763 Chinese logograms, but there are only four hundred different pronunciations for all these symbols. The syllable *yi*, for example, occurs 104 times in this character set, whereas only thirty logograms have unique pronunciations. Even when the tonal pronunciation is added (a manner of further delineating syllables spelled similarly in English), the ratio of unique sounds to symbols is still about 1:5 (an average of five characters have the same tonal syllabic pronunciation).

Now, take my name (shown first name then last name in the byline, unlike the Chinese tradition) as an example: There are twenty-one logograms with the pronunciation "feng" and twenty-five with the pronunciation "huang." If one wishes to retrieve a patient named <huang><feng>, as it would look in Chinese, one could use a Soundex-type algorithm to retrieve the name much more easily than without such a search strategy.

A second complication in the use of Chinese names is that the character set GB 2312-80 is not adequate to represent many Chinese names. Chinese parents often wish to use unusual historic logograms to name their children. When one cannot find the correct logogram for a Chinese name, then a Soundex reference might be used to search by sound for a given name.

There are two ways to implement a Chinese Soundex algorithm. First, one may set up a transform base for all available logograms. For instance, we might use a global with the following structure:

```
^DHFHZ(<"Chinese char>")="<Chinese phonetic spelling>"
```

which would create globals

```
   ...
   ^DHFHZ(<huang>="huang"
   ^DHFHZ(<feng>)="feng"
   ...
```

For the string <huang><feng> we could use a subroutine that would accept the key "huang-feng," then a Soundex algorithm could be applied to search for names of this form. This algorithm should be included in `SOU^DICM1`.

Or, one may use the same general process followed by `SOU^DICM1`. In the character set GB 2312-80, Chinese logograms are grouped into two sets called levels. The first level consists of 3,755 logograms arranged by their phonetic pronunciation (Hanyu Pinyin form). These characters can be processed by a modified algorithm in `SOU^DCM1` to generate a Soundex key. In other words, the pronunciation of each character would be stored in a table, grouped by those with similar pronunciations, as is implicit in the Soundex principle of ignoring vowels and grouping C, V, and B into one Soundex code. In this manner, all characters, even those in the second level of GB 2312-80, would be grouped into common phonetic pronunciations.

Besides the translation features listed above, a number of other minor issues arise, such as the operation of functions such as `DAYOFWEEK`, `LOWERCASE`, `MONTHNAME`, `REPLACE`, `UPPERCASE` and others. These problems are all soluble. A few minor problems of this type were dealt with in our translation, but they do not warrant explication in this article.

Results

We completed the Chinese translation of VA FileMan using the approach described in general terms here. All display and report messages are in Chinese, and all functions are retained in almost complete integrity. The interactions between the user and FileMan are all in Chinese with the exception of a few built-in English terms that are offered as options to simplify user interaction. (As noted earlier, however, Chinese equivalents are available if preferred.)

The Chinese VA FileMan has been implemented for regular use in the Patient Master File (ADT) in our hospital for several years. Some of its functions are especially welcomed by Chinese users. For example, the pointer type and cross reference are especially convenient for equating diagnosis strings in Chinese and their ICD-9 equivalents. This function and the data set function are considered particularly useful.

Although 8-bit ASCII-supported implementations of M are available in China today, there remain problems that require solution. Execution speed needs to be improved, and the internationalization process, already underway within the MUMPS Development Committee, needs to be accelerated so that standard forms of National Character Set definitions can be realized in the Chinese domain. Much work remains, and close international cooperation is needed to bring this effort to a speedy conclusion. *M*

Acknowledgements

Sincere thanks to Dr. Ichiro Wakai, Professor Richard F. Walters, and Professor William J. Harvey for their criticism, comments, and recommendations. We also recognize with thanks the pioneer work initiated by the MDCC-J in defining a Character Set Profile for Japanese that may serve as a model for definition of the Chinese equivalent.

Dr. Zheng Te began his involvement with M in the 1980s with a visit to Japan, and in 1988 he invited several Americans to China to introduce MUMPS. He pioneered the translation of FileMan in China, pursuading the president of the People's Hospital at Beijing Medical University to use FileMan in the management of medical records. Prior to his death in 1993, he was coauthoring this article with Dr. Huang.

Dr. Huang is a physician who now devotes his time to medical information management. He was a member of the Chinese delegation to the M Technology Association's Annual Meeting in 1993. He may be reached at People's Hospital, Second Clinical School of Medicine, Beijing Medical University, 42 Bai-Li-Shi Road, Beijing 1000044, People's Republic of China.

Endnotes

1. J.T. Huang and T.D. Huang, *An Introduction to Chinese, Japanese, and Korean Computing*, World Scientific Publishing, Singapore.
2. Walters, R.F., Ph.D., "Design of a Multilingual Workstation," *IEEE Computer*, 23:2 (1990) pp. 33–41.
3. MUMPS Development Committee (1994) American National Standard for Information Systems - Programming Languages - M, American National Standards Institute, Inc., draft document circulated for approval using canvass method, Summer 1994.