CX11: A Chinese Language Supporting System for X Window Environment

Xiong Zhiguo
Department of Computer Science
Peking University
Beijing, 100871 P. R. China

Abstract

Chinese language, unlike English and other Indo-European language family, deals with ideographic character input and output. The X Window System (X11R5 or earlier version) has the mechanism to output Hanzi (Chinese ideograph) through 2-byte encoding when the corresponding fonts are included, but it has no corresponding input feature. Because there are as many as 16000 most in use Hanzi, it need more than one keystroke to input a single Hanzi by the traditional English keyboard. This paper proposes a Hanzi input architecture on the X Window System and a Chinese language mechanism that can upgrade English based commercial X products or supply an application (X clients) programming interface for input Hanzi. CX11, our first implementation, is running on SCO ODT and other work-station. Its design principle and Implementation should be a typical one and can be applied as one of the general approach to internationalize and localize the X Window System.

Keyword: X Window System, Internationalization, Localization, Chinese Ideograph (Hanzi), Multibyte Encoding.

I. Introduction

Unlike English and other Indo-European language family, Chinese language deals with ideographic character (Hanzi) input and output. It has as many as 16000 most in use Hanzi, and needs at least two bytes to encode the Hanzi character set. Also, it needs more than one keystroke to input a single Hanzi by the traditional English keyboard. With the ever prevailing of computer application in China, the call for Chinese language support in system software and application become increasingly louder. But, there is no common or standard Chinese or internationalized language interface in the base of system (operating system or windowing environment), not to say the API and application user interface.

From the viewpoint of International language supporting, we can classify the software system into three catalogs: (1): the one developed only for English or Latin branch, no support for Asian language. Most software systems now in use belong to this catalog and are not easy or effective to update their basic language engine. (2): the one that only support 2-byte output with no corresponding input feature. (3): fully internationalized software system that separate the main processing part from native language specific part (such as hint, menu, and command). Although ISO and other standardization institute make much effort, and Microsoft (Window NT), Apple (System 7) and Novell (NetWare) will release their OS based on the Unicode which support multilingual (including Asian language), and other vendor will supply their individual system based on other encoding (such as IBM's Pink), Internationalizing process is not as quick as expected.

Thanks to X Window System's vendor-neutrality, OS independence and network transparency, extending X's mechanism to support multilingual input/output is one of the best shortcut solution for

internationalization until now. But the commercial X product is developed mainly for those whose language is English. Chinese and other Asian language (such as Kanji) can't be input directly in X environment, so, the Internationalization of X and the X-based GUI is an important concern for us and other non-English speaker.

In this paper, we will discuss CX11's Hanzi input architecture, schemes, and API. In CX11, a separate background process--Hanzi Input Manager (HZ-IM)--interact with X Server and X clients via Inter-Client Communication mechanism provided by X. Also we define a simple protocol between HZ-IM and application in order to let the application give instruction to HZ-IM. Based on this protocol, a Hanzi communication library, served as API, is constructed. After a unified font that combine Latin-1 and Hanzi is constructed, CX11 can also support existing commercial product (such as ODT's scoterm) as well. CX11's approach can applied to Kanji or other Asian language, but we only focus on Chinese in this paper.

II. Hanzi Input Schemes, Encoding and Input Manager

1. Input Schemes:

Like other ideographic language, Hanzi character set far exceeds what traditional keycap can represent and every single Hanzi 's input needs more than one key-stroke unless a very large specified keyboard is provided. What make things more complex is that there exists no simple structural pattern across the Chinese characters(5000 commonly used, another 11000 used occasionally). For each Hanzi, there may be a simplified glyph form used in mainland China or complex glyph form used in Taiwan and Hongkong. It is very difficult to design an easy-to-use and efficient Hanzi input scheme. Until now, there are at least 100 kinds of input methods(30 kinds in Taiwan alone). With the advance in the research of Chinese information process technology, there will be more input methods. Our CX11 should give support to all kinds of scheme and be extensible.

We can classify the current Chinese input schemes into 3 categories:

(1). Arbitrary Input Codes: with those schemes, every Hanzi associates with a unique digital number or alphabetic sequence without any meaning. Telegrame Code, Row-Column Code, GB Code belong to this class. Although it is almost impossible for every user to remember each Hanzi's input code, and thus not a good candidate, supporting these schemes are also necessary.

(2). Phonetic-Ideograph Conversion Codes: These schemes are based on character's pronunciation and are similar to Japanese kanar-kanji conversion. The most popular one is Pinyin which is constructed from Mandarin dialect. Although phonetic ambiguities in Pinyin can only resolved by secondly indexing or user's supplemental input and is not an efficient method, Pinyin is still the first choice for unprofessional user.

(3). Stroke Pattern Scheme: These are based on the form, the stroke pattern and the arrangement of strokes within a character. Generally speaking, there can be 30-50 common stroke patterns extracted from the whole Hanzi set. After encoding these stroke patterns on the keycap, this kind of scheme is the same as the second one except the occasional ambiguity. This scheme is potentially the best choice.

2. Input Manager's Conceptual Mode: the Finite State Automaton

Because there are so many kinds of input schemes and we must also process at least the English input while inputting Hanzi, we need design a separate program and construct a conceptual mode. In CX11, we call the program as Hanzi Input Manager(HZ-IM), the consideration of HZ-IM's implementation will be presented in Section 4.

In HZ-IM's heart there is a finite state automaton. When user press any key(or compound key), the entered character is not necessarily the character that is marked on keytop and is not directly input into the clients that acquires the input. HZ-IM will

intercept the entered character and put it as the automaton's input data. Different sequence of character leads to different state translation in automaton, and the IM will use the

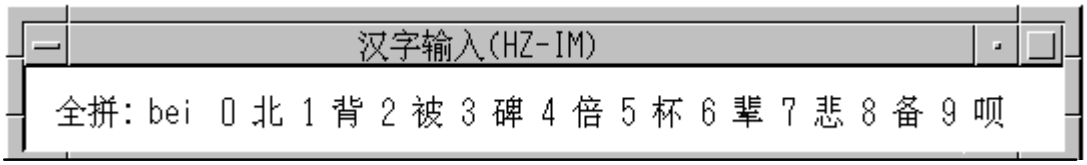

Figure 1. HZ-IM Displays a Conversion Window

automaton's last state and other state information to determine what the input sequence represents. In arbitrary input code category, the automaton's last state is unambiguous and IM will convert the sequence into a 2-byte Hanzi code and send it to the client. But in phonetic system, every input sequence may represent several(1 to 100) Hanzi, the automaton will pause at this intermediate state. After secondary indexing or end user's selection, the automaton will advance forward and arrives at an unambiguous state. In the stroke pattern catalogue, only a few sequences of input will lead the automaton into a intermediate state. No matter how vast the input scheme may be and how different their features may be, they all share the same automaton mechanism. As long as the dictionary that maps input sequence to Hanzi is supplied, HZ-IM can deal with multilingual and multi-scheme input.

3. Hanzi Encoding

It is obvious that we need at least 2 bytes to encode Chinese character set, but the problem is its actual implementation. In mainland China, GB2312-80 is the national standard Hanzi encoding system, but in Hongkong and Taiwan, Big5 is the most popular encoding system. GB2312-80(as many as 8800 Hanzi) uses 2 MSB byte to distinguish whether a byte pair is a Hanzi code or 2 ASCII codes while Big5(as many as 16000) use a MSB and a LSB. Extended UNIX Code(EUC) also use MSB to determine whether the character is made up of a single byte or multiple bytes.

In order to let CX11 support the simplified glyph form and complex glyph form used in different region, every Hanzi has a single code although it may have two forms. By default, we use GB2312-80 and Big5 code to extend the unencoded Hanzi that is in Big5 and not in GB2312-80, and we organize BDF font file according to GB2312-80 and this extension. Clients can notify HZ-IM to use which glyph form and which encoding system. HZ-IM will then load the corresponding font file, justify or convert the encoding system and constrain the glyph-specified input schemes.

III. Design Objectives and Architecture

In designing the CX11, we have the following objectives:
(1) OS independent. Because CX11 is programmed with Xlib, it is of course OS independent.
(2) Network transparency. This allows user to use remote input manager across network by X terminal or other workstation. As a X clients, CX11's input manager can be executed by remote X users when Chinese input is required.
(3) Input Manager being independent of other X clients. The input manager is designed as a daemon and it communicates with other clients with a simple protocol based on inter-client communication conversion manual.
(4) Supporting all kinds of encoding system(EUC, GB-2312-80 and Big5) and input schemes. As discussed above, our input manager's finite automaton mode can accept any kinds of input sequence.
(5) Extendibility and flexibility. CX11 can not only support Chinese Language, its design principle can be applied to other ideographic language as well.

In the architecture above, HZ-IM serves as a background daemon that can be activated by user through grabbed hot-key or by a X client or other application(such as UNIX's vi running on xterm through pseudo tty) when Hanzi input is needed. When invoked by X client, the client's communication library interact with HZ-IM and comply with a simple protocol. The Language specified dictionary is under the control of HZ-IM and can be dynamically loaded, unloaded or replaced by Input Manager or through X client's notification to HZ-IM.

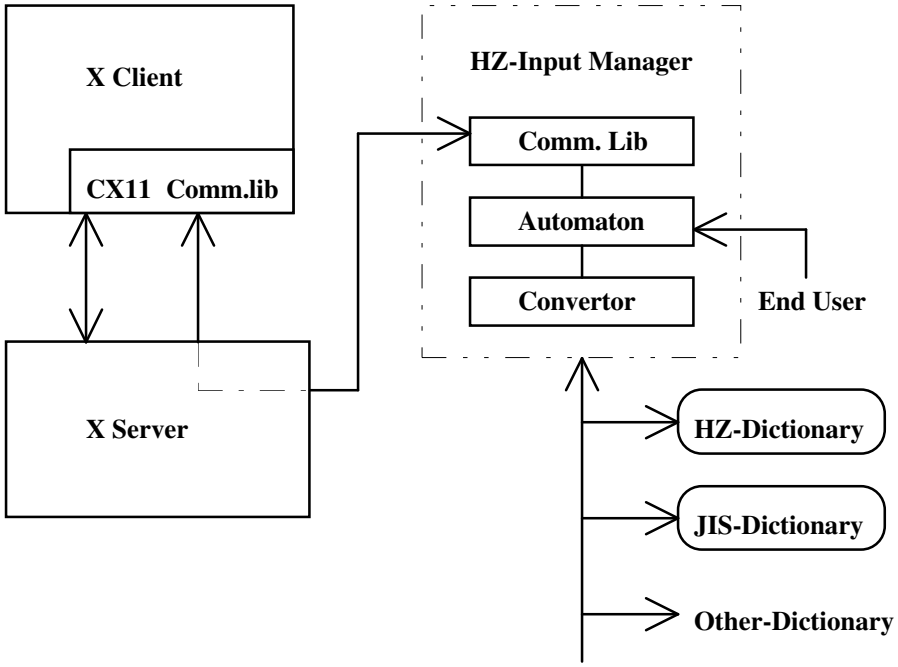

Figure 2. The Architecture of HZ-IM.

When HZ-IM is activated, every entered character is directed to input manager's finite automaton. The automaton will use the current loaded dictionary to convert the input sequence to a specified language code or report syntax error. The converted code will then be sent to the clients that acquires input. Our CX11's implementation proves that the architecture mode will not scarify the input respond both in network and local environment because of human's delay in the process of inputting. Because there is only one currently loaded dictionary, HZ-IM will not devour much memory.

IV. Implementation of HZ-IM: the Inter-Client Communication

1 Considerations in Implementation

From the architecture mode above, we know that HZ-IM is both a client and a server and it communicate with other X clients through X server and makes use of X server's mechanism to accomplish its input conversion. In actual implementation, we take the following issues into consideration:

How HZ-IM intercepts key event?

There may be three ways:

(1).Using XGrabKey and XSetInputFocus mechanism to set a hot-key to activate IM or have a property sent to IM for the notification of key-grabbing Input manager captures all the key events, reassigns X server's focus resource and displays a conversion window. IM may compete with window manager for events by this way, but carefully

programming will prevent this circumstance. CX11 uses this way.

(2). Putting an input-only window at the top of the client that acquires input and intercepting the key event for this window. When using this method, child windows of the clients might compete with input-only window for event. Because input-only window is transparent to user on the display, it is inconvenient for phonetic input scheme and for other schemes that have ambiguities.

(3). Extending X protocol to intercept key events. This is not a shortcut means.

How HZ-IM transmit the converted strings to the client?

(1) Transmitting events through XSendEvent. CX11 use this method.

(2) Setting properties. This may cause a synchronization problem between input manager and clients.

2. Communication By SendEvent

There are 3 kinds of data type sent by XSendEvent routine:

Table 1. SendEvent List

Data Type	Content	Event Type	Direction
char *	Converted Strings	ClientMessage	HZ-IM--->Client
int	Error Message	ClientMessage	HZ-IM--->Client
int	Key Event (raw)	KeyEvent	HZ-IM--->Client

(1)Converted Strings(Hanzi Word or Compound String):

Input manager sends converted strings to the clients when it has converted 2-byte code to a specified encoding system, or when it has got more than one Hanzi code(Hanzi word) by special input schemes, such as five-stroke method, or when it needs to send a string that contains more than one character type, such as Motif compound string.

(2) Error Message

An error message is reported by input manager to the clients. For the sake of minimizing the inter-client communication message, ClientMessage only contains the error number, the client who receives the error number can use XhzErrorMessage function to process the predefined error.

(3) Key Event

Because HZ-IM intercepts all masked key events by XGrabKey, it may conflict with some key operation originally defined for specified application, such as vi. For the key event that is not in the process of automaton or the key event that has more modifiers, HZ-IM will send it directly to the client.

3. Communication By Property

we have designed 3 kinds of Property:

Table 2. Property List

Property Name	Owner	Function
XHZ_IM_SETTING	Root Window	HZ-IM's current setting, used for querying
XHZ_CLIENT_CMD	Client's Window	used for sending command to HZ-IM such as connecting.
XHZ_CLIENT_SETTING	Client's Window	used for requesting HZ-IM change current setting

The 3 kinds of Property, together with 3 kinds of SendEvent, serves as CX11' s simple protocol(XHZ-Protocol) for the communication between input manager and other clients. The property is transparent to application programmer. Next section will define an API which is based on XHZ-Protocol.

V. Communication Library: the API

CX11 extends Xlib to let the client interacts with input manager and controls it in details through inter-client communication's Property mechanism. This prevents programmer from programming the CX11's XHZ protocols.

The API includes the following routines:

XhzIMInitialize

XhzIMinitialize requests that input manager be connected and initialized. It first tells HZ-IM about the client's window ID to let HZ-IM use it in XSendEvent, then specify the information about geometry and attributes of input conversion window. It may also specify the encoding system, font, input scheme, conversion dictionary and other parameters, or HZ-IM will use the default values.

XhzIMQuery

XhzIMQuery query HZ-IM's current setting in font, input scheme, conversion dictionary and encoding system.

XhzIMSetWin

XhzIMSetWin sets the conversion window's geometry and attributes and overrides previous setting.

XhzIMClose

XhzIMClose requests the disconnection with HZ-IM and free the resource if there is no other client waiting for HZ-IM's service.

XhzIMChangeScheme

XhzIMChangeScheme requests HZ-IM to use different input scheme and HZ-IM will load corresponding conversion dictionary and justify other resource. It is important that it is not permitted to change font or encoding system after calling XhzIMInitialize.

XhzRead

XhzRead requests HZ-IM input a converted string. Between the calling of XhzIMInitialize and XhzIMClose, all the key events will be intercepted by HZ-IM. HZ-IM will convert the input sequence based on the current setting and send the converted string to the client.

XhzErrorMessage

XhzErrorMessage processes the error reported by HZ-IM.

VI. Conclusion

Our implementation of CX11 has achieved design objectives, but the system is based on X11R4, we still have much to do to enhance our system for X11R5. Current CX11 is only running on SCO ODT(version 1.1 or 2.0) and is porting to SGI Indigo workstation. We expect it would be ported to other platform sooner.

Acknowledgment

I would thank Professor Dong for his encourgement, Mr. Xu Xi for his aiding in design and implementation, and Mr. Zhang Qian, Mr. Sun Honghui for their help in design and implementation. Special thanks go to Ms. Zhang Shuzhi. We also thank other members in our Computer Graphics Research Group.

Reference

1. Rober W. Scheifler and James Gettys, "X Window System", *ACM Transaction of Computer Graphics*, Vol. 5, no. 3, April 1986.

2. Rober W. Scheifler and James Gettys, *X Window System, Second Edition*, Digital Press, 1990.

3. David S. H. Rosenthal *Inter-Client Communication Conventions Manual*. X Window System, Version 11, Rel. 4.

4. ISO 2022: 1986 Information Processing-ISO 7-bit and 8-bit coded character sets - Code extension techniques, 1986.

5. ISO/IEC DIS 10646 Information technology - Universal Coded Character Set(UCS), ISO/IEC Draft International Standard, November 4, 1990.

6. Working Group on Internationalization, "Standards Update: Internationalization", in *An Update on the UniFroum Technical Committee, UniForum*, December 1989.

7. AT&T UNIX Software Operation Pacific, "Product Overview", in *Multi-National*

Language Supplement, Release 3.2, UNIX System V.

8. Masato Morisaki, Etsuo Kawada, Hiroshi Kuribayashi, Seiji Kuwari, and Masahiko Narita, Xjp System: An Internationalized Language Interface for the X Window System, *ACM sympoium on User Interface Software and Technology*, South Carolina, Nov. 1990.

Figure 3. A CX11's Hanzi window