4Q－3

The Worldwide Multilingual Computing (3): An Implementation of the Multilingual I/O TM/C System and Waseda X11

Tadao Tanaka‡, Yutaka Kataoka*, Kazutomo Uezono†, Tomoko Kataoka*, Hidejiro Daikokuya†, Toshio Oya†, Kenji Maruyama‡, Shoichiro Yamanishi† and Hiroyoshi Ohara†

* Centre for Informatics, Waseda University † School of Science and Engineering, Waseda University
‡ Research and Development, Japan Computer Corporation

1. Introduction

There were many trials of limited multilingual I/O and localization but they did not give enough focus on characters. Thus, the definitions and roles of essential informations and functions could not be defined clearly.

The first multilingual research based on all characters in the world was started at Waseda University and the essential informations and the functions for multilingual computing were discovered [1, 2]. And the *Meta Converter System* [3] and the basic components for the multilingual computing were developed [4].

By using the results, i.e., *Multilingual I/O TM/C System*, Locale related functions and X Window System were re-implemented as *WASEDA MLCE*. The X Waseda Window System (WX) re-implemented contains Multilingual OM, IM, Xlib and Xaw. WX can draw words from left to right, right to left, top to bottom and bottom to top, keeping correct character writing direction (Fig. 1).

Figure 1. Correct Multilingual Drawing

And all widgets can process such multilingual strings correctly. This multiple directional drawing functions pointed weakness of specifications of X Window System of both R5 and R6 [Talks 5, 6].

The re-implementations ensure backward compatibilities. But the re-implementations also revealed insufficiencies of specifications of C and OS [5, 6, 7, Talk 8].

2. Implementation of Multilingual I/O TM/C System

All mb, WC and Locale related functions can be implemented by the Multilingual I/O TM/C System that was realized by the *Meta Converter System* [3, 4]. The functions were stored in the *MLIOTMC library*. The Meta Converter System was modified to handle WC normalized as a *Character* [2, Talk 1, 2, 4]. The *Meta Converter Table Compiler* generates automaton bodies used by the Meta Converter System's functions [4]. The automaton bodies are optimized by basic conversion research [8]. The MLIOTMC library contains the Meta Converter System, and the library loads the automaton bodies into virtual memory space. Data files that describe conversion and WC bitfield information were arranged to satisfy the *Multi Locale Model* and the *Global IOTMC Model* – the compiler checks sufficiencies as much as possible.

For any *Graphic Character Set* and *Control Character Set*, any length of multiple designation sequence can be assigned. And beyond ISO 2022 [9], it is possible to accept and to generate functional sequence to extend the characters in the codesets. By the mechanism, specifying one specific glyph of *Perso-Arabic* scripts could be satisfied.

3. The Re-Implementation of X Window System

The *Output Method* (OM) was developed to draw correctly [8]. The OM has a role to convert from mb/WC to sequence of a pair of a logical font name and index to a glyph in it. During the conversion, 1 of 16 glyphs is selected [2]. The conversion rules are also stored in a file and compiled to automaton bodies which are loaded when OM is invoked. Also OM performs to re-order words according to writing direction [8]. The sequence generated by OM is processed by the *Quad Directional Drawing Module* (QDDM) which associates real font file and draw 4 directions [2]. Since X Window System can draw only left to right, the QDDM emulates other direction drawings by X's basic functions. Thus, OM is independent from windowing systems except for QDDM and functions returning character and cursor locations. Drawing functions in libX11 were replaced by the functions with the same name provided by OM's top layer – specification dependent layer [2].

The *Input Method* (IM) must be connected to libX11. But old IM protocols are completely not sufficient to control IM's behavior when drawing direction is changed. Thus, new IM library (MLIMlib) is sufficient to communicate with the new IM [Talk 6]. The MLIMlib receives character sequences via *Interprocess Communication Assisting Functions* (ICAF) in the Meta Converter System after encoding scheme of the string is converted to given state and/or locale. The libX11 can

communicate with Multilingual Waseda IM [Talk 6].

4. Re-Implementation of Athena Widget

Widgets relating text processing were all modified to be multilingual; widgets in Simple Widget, Menu Widget and Text Widget. All of them satisfy both *Multi-Locale Model* and *Global IOTMC Model*. Thus, each widget may have one locale or Global (ISO 2022), and the locale can be changed interactively (Fig. 2).

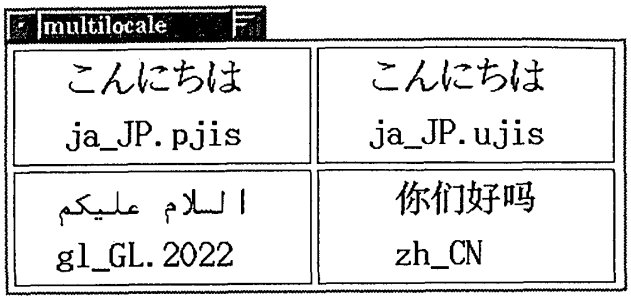

Figure 2. Multiple locales per application

The widgets support all drawing directions and all line progress directions in any model.

Since the process unit in the widgets is a character, all strings coded in mb are converted to corresponding strings in WC [10, Talk 5]. During the conversion, default information to specify writing direction, origin of line and direction of line progress are added into the bitfield of WC. Also, information specifying permissions for the line break is embedded into the bitfield. The information to determine origin of line and origin of line progress can be changed interactively. The information above also can be specified by the resource file.

The cursor shows direction of a word and direction of a character to the next word and to the next character. Since memory image and display image do not match, the role of the cursor is essential [Talk 5].

By using of a pointing device, cut and paste can be performed among different locales. Note that interprocess communication is performed according to *ICCCM* that limits ISO 2022 encoding schemes. When locales are different between a sender and a recipient, only characters in *Graphic Character Set* (GCS) shared by both locales are displayed. In this case, two ways can be chosen by users; 1) removing characters in unshared GCS and 2) retaining characters in unshared GCS. The ICAF convert the *Compound Text* encoding in ICCCM to given locale. To perform the Compound Text communication, *Cut Buffer* must hold any octet (00/00 to 15/15) by change of *X String* redefinition.

WC strings in the widget are converted into mb specified by given locales or ISO 2022 for Global IOTMC Model.

5. Conclusion and Remarks

The basic multilingual computing environment was implemented by Multilingual I/O TM/C and the implementation can run correctly. The performance of the X Waseda Window System is almost the same as X11R5 by the optimal algorithm and implementation. By the implementation, specifications of POSIX and X Window System could be pointed where problems and insufficiencies occurred. Also extensions of the specification in the implementation could clearly define what specification should be taken. To choose the best user interface specifications for the multilingual processing beyond current specifications, a lot of application softwares are implemented on the WASEDA MLCE.

References

[1] Kataoka, Y. et al., A model for Input and Output of Multilingual text in a windowing environment, ACM Transactions on Information Systems, Vol. 10, No. 4, October 1992, pp 438-451.

[2] Kataoka, Y. et al., 1995. Codeset Independent Full Multilingual Operating System: Principles, Model and Optimal Architecture, IPSJ SIG System Software & Operating System, 68-4, pp. 25-32.

[3] Kataoka, Y., et al., Multilingual I/O and Text Manipulation System(1): The Total Design of the Generalized System based on the World's Writing Scripts and Code Sets, Proceedings of the 49th General Meeting of IPSJ, Vol. 3, September 1994, pp 299-300.

[4] Tanaka, T., et al., Multilingual I/O and Text Manipulation System(4): The Optimal Data Format Converter to/from MB/WC/TMC, Proceedings of the 49th General Meeting of IPSJ, Vol. 3, September 1994, pp 305-306.

[5] ISO/IEC 9899: 1990, Programming language C.

[6] ISO/IEC 9899: 1990/DAM 3, *Draft* Amendment 1:1994 (E), Programming languages – C AMENDMENT 1: C Integrity.

[7] ISO/IEC 9945-1: 1990, Information technology – Portable Operating System Interface (POSIX) Part 1: System Application Program Interface (API) [C Language].

[8] Uezono, K. et al., Multilingual I/O and Text Manipulation System (2): The Structure of the Output Method Drawing the World's Writing Scripts beyond ISO 2022, Proceedings of the 49th General Meeting of IPSJ, Vol. 3, September 1994, pp 301-302.

[9] ISO/IEC 2022: 1986, Information processing – 7-bit and 8-bit coded character sets – Code extension techniques.

[10] Kataoka, T. et al., Multilingual I/O and Text Manipulation System (3): Extracting the Essential Informations from World's Writing Scripts for Designing TMC and for the Generalizing Text Manipulation, Proceedings of the 49th General Meeting of IPSJ, Vol. 3, September 1994, pp 303-304.