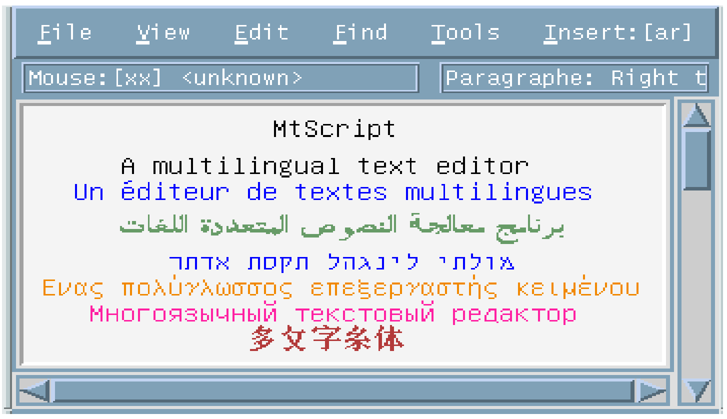


MtScript

A multilingual text editor
Un éditeur de textes multilingues
برنامج معالجة النصوص المتعددة اللغات
אולתי לינגהל תקסן אדתך
Ενας πολύγλωσσος επεξεργαστής κειμένου
Многоязычный текстовый редактор
多文字条体

Malek Boualem & Stéphane Harié

Laboratoire Parole et Langage, CNRS & Université de Provence
29, Avenue Robert Schuman, 13621 Aix-en-Provence Cedex-1, France
E-mail: mtscript@lpl.univ-aix.fr

MtScript is the winner of the 1996 CNRS/ANVIE prize for the scientific research industrial valorization

Abstract

This paper describes the multilingual text editor **MtScript**[1] developed in the framework of the MULTEXT project. **MtScript** enables the use of many different writing systems in the same document (Latin, Arabic, Cyrillic, Hebrew, Chinese, Japanese, etc.). Editing functions enable the insertion or deletion of text zones even if they have opposite writing directions. In addition, the languages in the text can be marked, customized keyboard input rules can be associated with each language and different character coding systems (one or two bytes) can be combined. **MtScript** is based on a portable environment (Tcl/Tk). **MtScript.1.1** version has been developed under Unix/X-Windows (Solaris, Linux systems) and other versions are planned to be ported to the Windows and Macintosh environments. The current 1.1 version presents several limits that will be fixed in future versions, such as the justification of bi-directional texts, printing support, and text import/export support. Future versions will use SGML and TEI norms, which offer ways of encoding multilingual texts and are to a large extent meant for interchange.

Keywords

Multilingual text, document, character, keyboard input, coding standards, editing, textual data interchange.

Résumé

Cet article présente l'éditeur de textes multilingues **MtScript** développé dans le contexte du projet MULTEXT. **MtScript** permet de mixer nombreux types d'écritures dans un même document (latin, arabe, cyrillique, grec, hébreu, chinois, japonais, etc.). Ses fonctions d'édition permettent d'insérer ou de supprimer des zones de texte même en écritures en sens opposés. De plus, **MtScript** permet de marquer les langues utilisées dans un texte multilingue et de leur associer des règles de saisie au clavier et de traiter différents types de codage des caractères (un ou deux octets). Enfin, **MtScript** a été développé dans un environnement portable (Tcl/Tk). La version **MtScript.1.1** a été développée sous Unix/X-Windows (systèmes Solaris, Linux) et des versions ultérieures seront portées sur les environnements Windows et Macintosh. Toutefois, la version actuelle 1.1 a des limites qui seront traitées dans des versions futures telles que la justification des textes bi-directionnels, l'impression et le format d'échanges de textes malgré que les caractères sont représentés dans des codes standards. Les prochaines versions utliserons les normes SGML et TEI qui offrent des méthodes pour le codage et l'échange des textes multilingues.

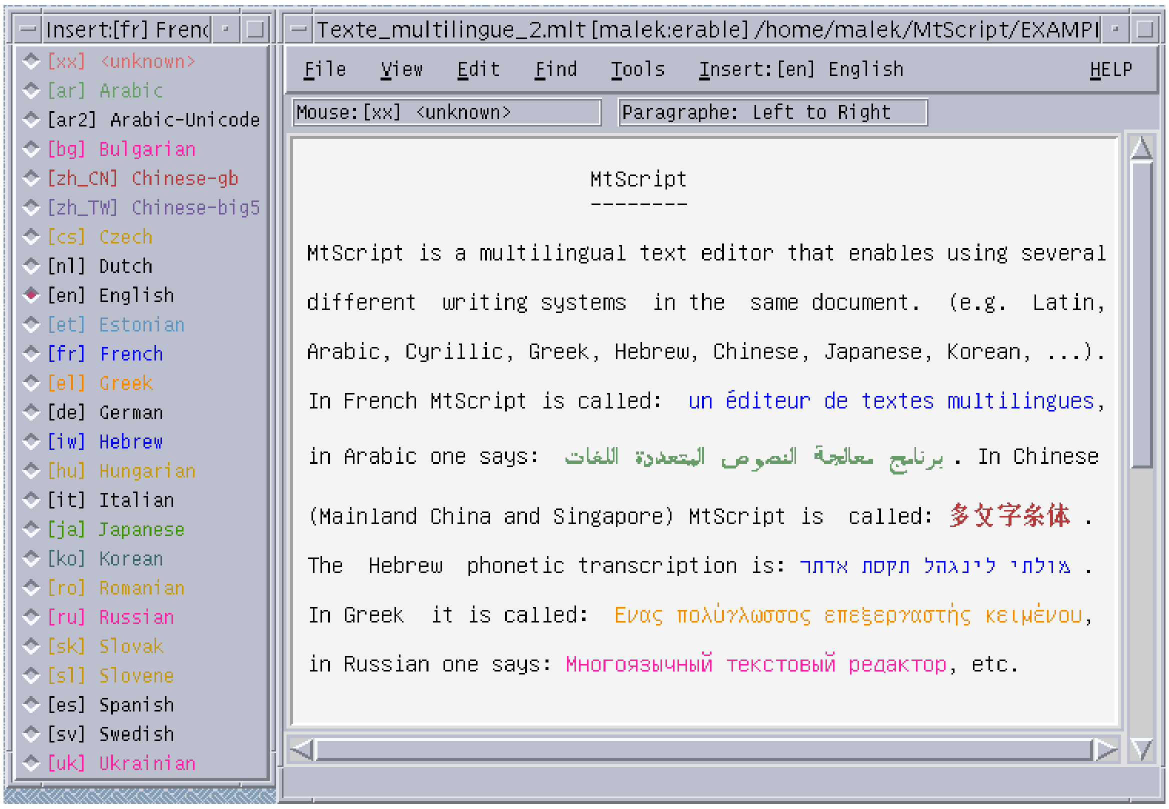
---

[1] **MtScript.1.1** for Solaris and Linux systems is freely available in compiled form and can be downloaded at the URL: **http://www.lpl.univ-aix.fr/projects/multext/MtScript/**

Mots-clés
Texte multilingue, document, caractère, saisie, norme de codage, édition, échange de données textuelles.

1. Introduction

In a previous paper [BOUA95a] we outlined difficulties in the multilingual text editor's design and we mentioned that although solutions for European languages already exist, the processing of other language is still at a conceptual stage. We presented the prototype of the **TE** multilingual text editor [BOUA90] which we integrated into a machine translation system from French to Arabic [BOUA93]. The **TE** editor showed up weaknesses in the character and document coding, incompatibilities in exchanging texts, and problems with non-portable environments. In this paper we present the **MtScript** multilingual text editor (see figure 1) that we developed within the context of the MULTEXT[2] project. **MtScript** allows numerous languages to be mixed within the same document, even in bi-directional writing. **MtScript** allows the user to identify the languages used within multilingual texts and to associate them with keyboard specifications and writing rules. Moreover **MtScript** was developed in a portable environment (Tcl/Tk) and is based on both single-byte and multiple-byte international character-coding standards.


Insert:[fr] Frenc
Texte_multilingue_2.mlt [malek:erable] /home/malek/MtScript/EXAMPI
File
View
Edit
Find
Tools
Insert:[en] English
HELP

```
[xx] <unknown>
[ar] Arabic
[ar2] Arabic-Unicode
[bg] Bulgarian
[zh_CN] Chinese-gb
[zh_TW] Chinese-big5
[cs] Czech
[nl] Dutch
[en] English
[et] Estonian
[fr] French
[el] Greek
[de] German
[iw] Hebrew
[hu] Hungarian
[it] Italian
[ja] Japanese
[ko] Korean
[ro] Romanian
[ru] Russian
[sk] Slovak
[sl] Slovene
[es] Spanish
[sv] Swedish
[uk] Ukrainian
```

Mouse: [xx] <unknown>   Paragraphe: Left to Right

```
                 MtScript
                 --------

MtScript is a multilingual text editor that enables using several
different  writing systems  in the  same document.  (e.g.  Latin,
Arabic, Cyrillic, Greek, Hebrew, Chinese, Japanese, Korean, ...).
In French MtScript is called:  un éditeur de textes multilingues,
in Arabic one says:  برنامج معالجة النصوص المتعددة اللغات . In Chinese
(Mainland China and Singapore) MtScript is  called: 多文字条体 .
The  Hebrew  phonetic transcription is: מולתי לינגהל תקסת אדתר .
In Greek  it is called:  Ενας πολύγλωσσος επεξεργαστής κειμένου,
in Russian one says: Многоязычный текстовый редактор, etc.
```

---

[2] MULTEXT is the generic name of a set of projects coordinated by the CNRS "Parole et Langage" Laboratory: LRE-MULTEXT (*Linguistic Research and Engineering* Program), MULTEXT-EAST (*Copernicus* Program), MULTEXT-CATALOC (Program of *Langues Régionales et Minoritaires* de la DGXXII), ALAF Research Shared Action (Alignement of African and French Languages, AUPELF•UREF). MULTEXT aims to build standard methods for linguistic data representation and to develop language processing tools for about fifteen languages.

Figure 1. A view of MtScript editor

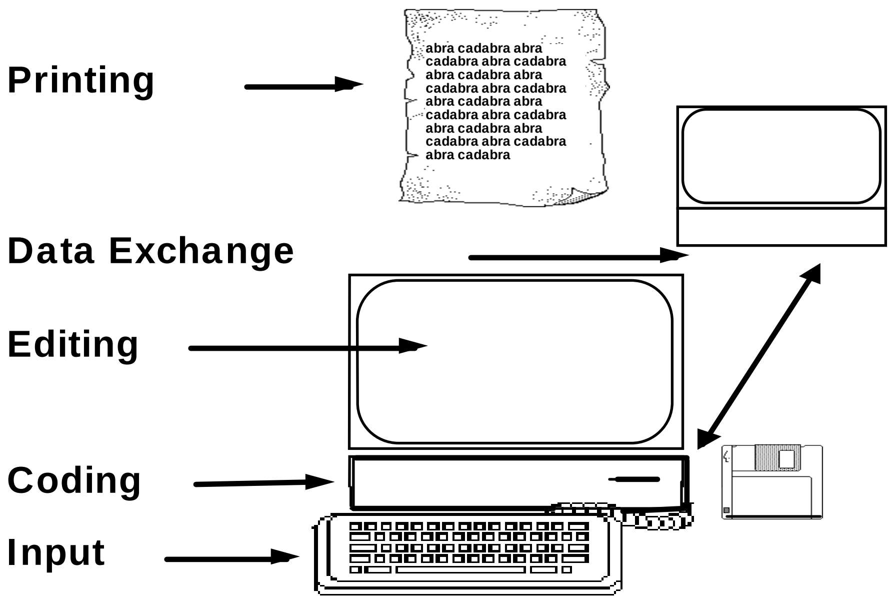
2. Conceptual difficulties of multilingual editing tools

Users of more and more applications now require multilingual text-editing tools, including word processors, database creation and management systems, and desktop publishing systems. In the area of automatic or machine-aided translation, multilingual text editors are a basic tool for pre-editing source text and post-editing target text [BENT91]. Another new area where multilingual text editors could be of great use is that of internationalization and localization of software and associated documentation for use in a multi-cultural environment. These areas were born as a direct effect of the emergence of new technology and the globalization of the Information Technology market. Many organizations and projects work to one extent or another within these areas (LRE Glossasoft project, CEC, CEN, Esprit, Eureka, Internet, JSA, Linux International, Unicode, TEI, etc.).

The processing of languages not based on the Roman alphabet poses a number of difficulties. For example:

- Arabic is written from right to left.
- Chinese contains thousands of ideograms, which obstructs a one byte coding.
- In Thai and other Indian languages, the sequence of characters does not correspond to its phonetic equivalent and one character may even be drawn encircling others.
- In Korean, characters are fused to make syllables.
- ...

Multilingual text-editing implementational difficulties occur on several levels: keyboard input, coding, editing, printing and data exchange (see figure 2).



Multilingual text-editing implementational difficulties

Figure 2.

2.1. Keyboard input

Though many keyboards represent only ASCII graphic characters (or ISO 646), certain localized (adapted) keyboards may also include keys for special or accented characters. For example, French keyboards generally feature keys corresponding to **"à ç é è ù"** accented characters, while characters which contain circumflexes or dieresis **"ê ï"** are input by two successive key presses. In addition, there is generally no single key on a French keyboard that allows one to produce characters that exist in other European languages, such as **"ñ"** or **"ò"**. In a broad multilingual context one could scarcely begin to imagine a keyboard that contains all possible characters. The inclusion of such languages as Chinese (with more than 6000 ideograms) or Arabic (approximately 4 sets of 28 letters and 10 vowels) requires the definition of specific keyboard input programs. Solutions proposed by computer manufacturers are very heterogenous. Theoretically there exists a standard input method for keyboards with 48 keys (ISO/IEC 9995-3), at least for the Roman alphabet, but it is rarely used. A number of keyboard input methods for the ISO 10646 characters was recently proposed [LABO95] using hexadecimal codes or composition. But these keyboard input methods always require the user to know and memorize a huge number of codes and it is necessary to develop more intuitive keyboard methods and, if possible, reduce the number of key presses by the user.

2.2. Coding

2.2.1. Character coding

Computer manufacturers and software developers use numerous specific and non-compatible character codes *(MS-Windows character set for Western Europe MS CP1252, DEC Multinational character set, International IBM PC character set IBM CP850, Macintosh Extended Roman character set, Hewlett-Packard ROMAN8, etc.).* Meanwhile other character coding norms have been standardized on an international level and are already used in some environments. In particular, the ISO 8859 code proposes a standard character set for the Roman, Cyrillic, Greek, Arabic, and Hebrew alphabets. More recently (1993) the ISO 10646 *(Universal multiple-octet coded character set or UCS)* proposed a universal character set including all the character sets of ISO 8859 as well as those for Chinese, Korean, Japanese, the International Phonetic Alphabet, etc. In its present form (ISO 10646-1), the UCS uses a 16-bit code (UNICODE) which will be extended to a 32-bit one in future editions, thus permitting an effectively unlimited coding of characters [JAMG95]. However, existing environments are not yet ready to implement character sets on multiple-octet code, even though the situation is rapidly improving (e.g. Windows-NT, AT&T Bell Plan 9 and Apple QuickDraw GX). Moreover SGML entities have been defined for encoding the characters of many different languages. SGML is beeing a standard for the multilingual document interchange.

2.2.2. Writing systems coding

In a multilingual text it is necessary to code not only individual characters but also scripts (Latin, Semitic, ...) and languages. In the case of a one-octet-based coding (e.g. ISO 8859-* character sets), it is necessary to mark the change from one set to another (e.g changing from Greek to Cyrillic). This can be done using a code such as that proposed in the ISO 2022, which includes escape sequences (<SI> (shift in) and <SO> (shift out)) that encode a transition between the "main" and the "complementary" sets. However these techniques are limited and many difficulties can arise, especially when a single document includes one-byte (e.g., ISO 8859-*) and two-byte (e.g., GB-2312-80 or BIG-5-0 for Chinese, JISX0208-1983-0 for Japanese or KSC5601-1987-0 for Korean) characters. The UCS inventory solves one part of the problem by combining all these character sets into a single set, since it is no longer necessary to implement a means for switching between character sets. However the problem is not totally resolved because UCS does not explicitly encode some features of the character sets such as the writing direction (although bidirectional protocols have been proposed by the Unicode Consortium). Moreover language tagging is needed not only to indicate writing direction, but also to control hyphenation, ligation, font selection and character/glyph mapping.

2.2.3. Language coding

Linguistic processing of a multilingual text (segmentation, morphological and lexical analysis, etc.) requires the identification of the languages therein. Recognizing the character set or the writing system does not suffice to identify the language in which a portion of text is written: a document encoded in ISO 8059-1 could equally well be written in French, English, Spanish or even a combination of these languages.

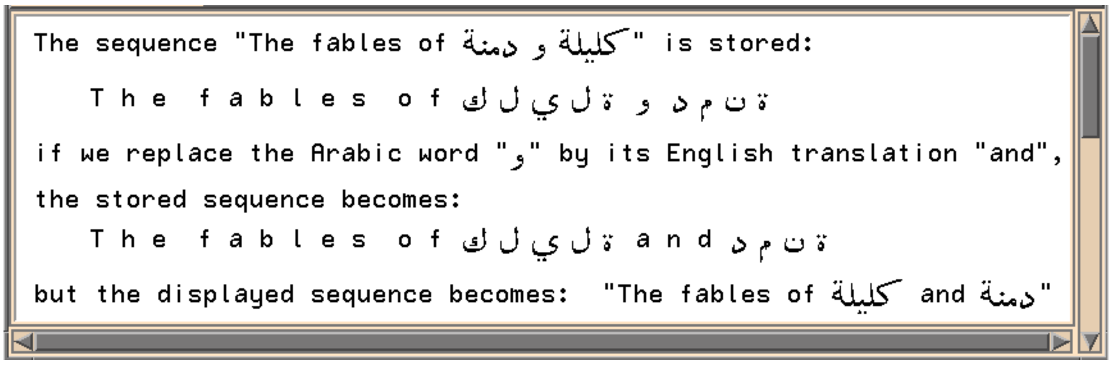
Norms for coding the names of languages exist:

- ISO 639-1988: 2 alphabetic letters code for about 140 languages (e.g "en" for English, "fr" for French, etc.).
- ISO 639-2: 3 alphabetic letters code, alpha-3, is currently in development (e.g "eng" for English, "fra" for French, etc.).

However, in the internal code of a document, these codes cannot be used such as they are. At this time there is no established standard method for escape sequences which would permit the representation of the change from one language to another, although it has been proposed that one use the ISO/IEC 6429 set of control sequence codes with a numeric conversion of the above alphabetic codes [LANG93]. Language markup is also currently being defined in the SGML/HTML standard used by the World Wide Web [YERG95].

2.3. Editing

The majority of languages are written horizontally from left to right. Some languages, such as Arabic or Hebrew, are written from right to left. Other languages, such as Chinese or Japanese, can even be written from top to bottom (especially in ancient texts). As a consequence, the co-existence of languages in the same document, and particularly on the same line of the text, poses huge problems when inserting or deleting text zones. The example in figure 3 shows that it is often necessary to rearrange words to maintain the semantic coherence of a sentence.



```
The sequence "The fables of كليلة و دمنة" is stored:
   T h e   f a b l e s   o f ك ل ي ل ة  و  د م ن ة
if we replace the Arabic word "و" by its English translation "and",
the stored sequence becomes:
   T h e   f a b l e s   o f ك ل ي ل ة  a n d  د م ن ة
but the displayed sequence becomes:  "The fables of كليلة and دمنة"
```

Figure 3. Editing aspects in a multilingual text

2.4. Printing

Printing multilingual texts suffers most obviously from the lack of printer fonts (essentially PostScript fonts). Many PostScript fonts are now available (freely or not) for Roman characters, but only a few fonts have been developed for the other character sets. Significant new efforts in this area include the OMEGA project activities [YHJP95] for multilingual TeX and the works of C.Bigelow and K.Holmes [CBKH95] in designing a UNICODE font *Unicode Lucida Sans* for editing and printing multilingual electronic documents.

2.5. Data exchange

With the rapid growth in the use of the Internet, the electronic transfer of multilingual documents is becoming more and more necessary. Until recently, only one part of the standard invariant characters of the ISO 646-IRV (ASCII) could allow a non-corrupted electronical text exchange, and multilingual documents could be transmitted safely only with the assistance of coding utilities such as *UUENCODE* and *BINHEX*. However the situation is improving: standards have been adopted on the Internet which allow the transfer of 8-bit characters without corruption in the TCP/IP protocol (for example, applications such *TELNET* and *FTP* are "8-bit clean"). In addition, the MIME norm *(Multi-purpose Internet Mail Extension: RFC-1521 and RFC-1522)* allows uninterrupted data transfer in any circumstance by compressing and decompressing the files. Moreover the emerging general standard for text data interchange is SGML (and for certain areas TEI), although these standards are not yet universal and some transfer problems persist. In saying this, one must point out that the current guarantee for data transferral without corruption does not extend to the transfer of multilingual data. It is necessary that both parties involved in the transfer, the sender and the receiver, have the same systems of encoding characters, documents, languages and writing systems.

3. Existing tools

The ground work for producing multilingual text editors has frequently been carried out under the form of independent experimental studies, often leading to incompatible products which are difficult to use and do not conform to coding norms. In addition to this, the proposed solutions often concern only languages using the Roman alphabet and cannot be adapted to other families of languages.

Among existing multilingual text editing tools we may mention the "Universal Word" developed by Wysiwyg Corporation and the "Wintext" program developed by Winsoft in the Apple Macintosh environment. These allow the mixture of several languages in the same document, even those written in an opposite direction. In the PC-Windows environment we may mention the Microsoft word processing software "Word", which uses the multilingual interfaces TwinLink and TwinBridge. On workstations (including some non-UNIX systems) the $\TeX$ type-setting language and its multilingual extensions (ArabTeX, etc.) is used primarily in scientific applications. Various $\TeX$ fonts for different languages have been designed but this system has the disadvantage of not incorporating a WYSIWYG editor, at least on some platforms. The OMEGA project includes a number of $\TeX$ extensions designed to improve multilingual text processing. It uses the ISO-10646/UNICODE standard code with conversion mechanisms for other standard codes. Powerful algrithms allow the interpretation of the composition and the transliteration of non-Latin characters (user interface), the handling of different character codes (information exchange), and the generation of correct character graphical components as ligatures (typography). Meanwhile efforts are under way to make the widely-used GNU Emacs editor suitable for languages other than English (MULE editor). At first it seemed worthwhile for us to extend Emacs with improved multilingual support. But due to that Emacs is not a WYSIWYG editor and due to the advantages of employing Tcl/Tk facilities for developing text processing tools and for adapting them to other new multilingual applications (WWW, etc.), we developed MtScript under that environment.

More recent work in the multilingual domain includes the CRL Laboratory activities [CRL96] in designing tools for various domains (multilingual machine translation, text retrieval, multilingual dictionnaries, etc.), the Accent company activities [ACC96] in designing multilingual WWW browsers, and the Technion institute activities in designing editing tools for bi-directional texts [BERRY92].

Unlike the previously mentionned multilingual text editors, **MtScript** is a WYSIWYG multilingual text editor (the only one available on SUN Workstations), freely distributed on the Internet, and is based on an environment (Tcl/Tk) that is parametrizable, evolutive, and portable (Unix, Windows and Macintosh).

4. Description of the MtScript editor

4.1. Main features of MtScript

The **MtScript** editor was developed in the Tcl/Tk environment, which provides the following advantages over other existing multilingual text editors:

• Tcl: script language (the commands are interpreted interactively),

• manipulation of textual data (characters, fonts, words, etc.),

• ability to define character attributes,

• ability to manage X-Windows events (mouse, keyboard, etc.),

• bitmap control,

• easy to use (X-Windows, buttons, icons, etc.),

• portability to other environments (Windows, Macintosh, etc.).

MtScript.1.1 currently runs under Unix/X-Windows. Unix has the ‘locale’ for multilingual support and X-Windows has ‘resources’ for tuning the visual attributes of applications (fonts, colours, sizes, etc.). To make **MtScript** parametrizable on the system level, we link the character sets to the fonts via fonts.alias resource files. This allows the user to redefine character attributes such as sizes, colours, etc. and facilitates the portability of the software to other environments.

MtScript is a text editor including most of the characteristics of standard monolingual editors, and it allows:

• mixing of left-to-right and right-to-left writing on the same line of text,

• recognition of the language used in a given piece of text,

• insertion/deletion of characters regardless of the direction in which the text is written,

• text editing functions: copy, cut, paste, etc.

MtScript is independent of any language. The languages are considered external parameters of the program and are represented by writing rules files and character fonts. The expansion of the editor to include a new language simply requires the inclusion of new character fonts and writing rules.

4.2. Internal representation

In its current version, **MtScript** handles the following character sets:

- iso8859-1, 2, 3 and 4 (Roman Alphabet)
- iso8859-5 (Cyrillic)
- iso8859-6 (Arabic)
- iso8859-7 (Greek)
- iso8859-8 (Hebrew)
- gb2312-80 and big5-0 (Chinese)
- jisx0208-1983-0 (Japanese)
- ksc5601-1987-0 (Korean)

In future versions, we hope to adopt the UCS set (ISO 10646), which includes other writing systems and a large number of characters absent from the norms presently included (for example, the conjoined symbols "œ" and "Œ", which are considered in French to be distinct from their component parts).

Labelling of characters and languages in a multilingual text is executed with reference to a "style file" associated with each multilingual text and containing values of the character properties. These properties describe for each piece of text: languages, fonts, character sets, style, tabulations, height and colour of characters. These are associated with a fixed position in the text and expressed by line numbers and character numbers. Figure 4 gives a partial internal representation of the "style file" associated to the text in figure 1. We are currently developing an SGML/HTML exchange format which uses the <LANG> tag proposed by the HTML.3.0 norm.

```
{mtscript_version 1.2}
{default_style
  { -width 80}{ -height 40}
  { -tabs {52.0 104.0 156.0 208.0 260.0 312.0 364.0 416.0 468.0 520.0 572.0 624.0 676.0 728.0 780.0}}
  { -wrap char}}
{newpage}
...
{ar {-foreground PaleGreen4 -font arabic} {13.22 13.63}}
{ar2 {-foreground Black -font arabic_unicode} {}}
{zh_CN {-foreground brown -font gb2312_1980} {15.53 15.63}}
{zh_TW {-foreground MediumPurple4 -font big5_0} {}}
{en {-foreground black -font iso_8859_1} {1.0 11.32 11.65 13.22 13.63 15.53 15.63 17.41 17.63 19.26 19.64 21.22
21.53 22.0 37.0 41.0}}
{fr {-foreground black -font iso_8859_1} {11.32 11.65 22.0 37.0}}
{el {-foreground DarkOrange2 -font iso_8859_7} {19.26 19.64}}
```

```
{iw {-foreground blue -font iso_8859_8} {17.41 17.63}}
{hu {-foreground DarkGoldenrod -font iso_8859_2} {}}
{ja {-foreground ForestGreen -font jisx0208_1983_0} {}}
{ko {-foreground DarkSlateGray -font ksc5601_1987_0} {}}
{ru {-foreground DeepPink1 -font iso_8859_5} {21.22 21.53}}
...
```

Figure 4. Extract of the style file associated with the multilingual text in the Figure 1

4.3. Multilingual text keyboard input

MtScript uses keyboard input methods based on those characters that are found on almost all keyboards, i.e., those of ISO 646-IRV. There are two types of keyboard input methods or programs:

- Alphabetical keyboard input program for alphabetical languages (English, French, Arabic, Hebrew, Russian, etc).
- Phonetic keyboard input program for ideogram-based (and phoneme-based) languages (Chinese, International Phonetic Alphabet, etc.). This program is included in the **MtScript.2.0** version (nearly available).

4.3.1. Alphabetical keyboard input program

A single alphabetical program is used for inputting texts in all alphabetical languages. Of course this supposes that using **MtScript** to input non-Roman characters requires localised keyboards based on the corresponding language transliteration standards. The keyboard input program uses separate "writing rules" and "transliteration rules" files for each language (figure 5).

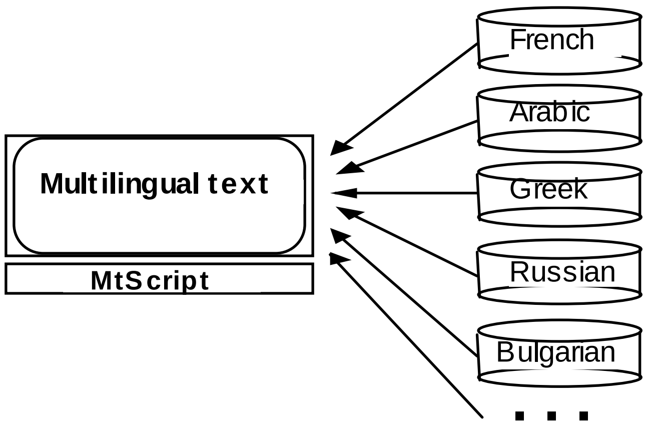


Figure 5. Language writing rules

Each writing rules file contains:

- a classification of characters according to their behaviour, e.g.,
  - lowercase letters which may have an accent,
  - lowercase letters which may not have an accent,

- uppercase letters which may have an accent,
- standard characters used for typing accents,
- numbers (e.g. in Arabic numbers are edited in an insertion mode),
- etc.

• a set of writing rules e.g.

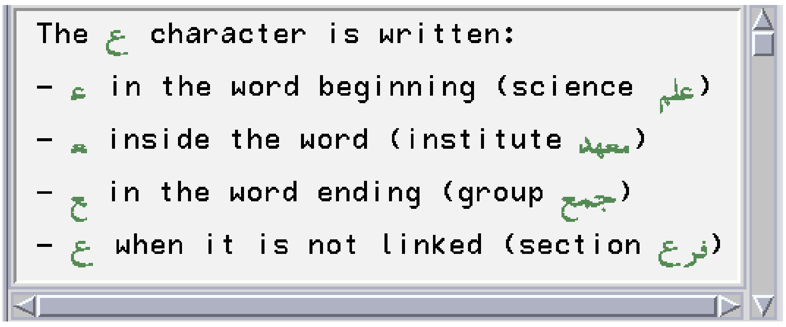
- French:	**e + ' => é** ; **c + , => ç** ; **e + ESC + ' => e’** ; etc.
- Greek:	**σ** (beginning and middle of a word), **ς** (word ending)
- German:	**s + s => ß** ; **s + ESC + s => ss** ; etc.

The writing rules for each language are expressed in an intuitive formalism based on finite state automata mechanism (we are planning to convert that formalism to use a more intuitive one, based on regular/symbolic expressions). The writing rules are then compiled and converted into internal tables usable by the keyboard input programs. A default set of rules is defined for each language, but individual rules can be redefined by the user, according to specific needs or preferences or to suit the particular specifications of certain keyboards. The default rules are based on the following principles:

• **Characters with accents** require two key presses, according to the rules for the language. Thus, in the default rules for French, **e+'** gives **é**, but for the same combination in English, it gives **e'**. To produce **e'** in French (a combination which is far less frequent then **é**) one uses the escape sequence **e + ESC + ' => e'**

• **Non-Roman characters** are typed following, as closely as possible, the conventions of the specific language and standard transliteration norms (where they exist). The transliteration tables are included in a "transliteration rules" file associated with each language (e.g. ISO 233-1984/1993 for Arabic, ISO 259-1984 for Hebrew, ISO/R 843-1968 for Greek, etc.). Thus, one types **a** for Greek **α**, **b** for Greek **β**, **s** for Arabic **س** , etc.

• **Variant forms**, such as **σ** and **ς** (Greek), **ß** (German), or the positional variants of Arabic letters, are generated automatically according to their context, so that the user does not have to intervene. The case of Arabic is particularly interesting [BOUA95b]: the alphabet contains 28 letters, the majority of which can be written in 4 different forms, according to their position in a word (figure 6). **MtScript** handles the positional variants of characters even during insertion or deletion operations.



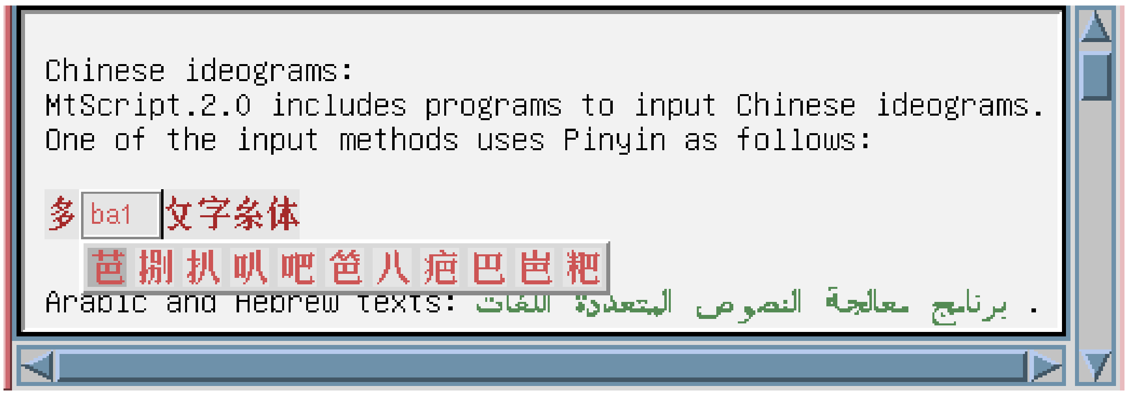
```
The ع character is written:
- عـ in the word beginning (science علم)
- ـعـ inside the word (institute معهد)
- ـع in the word ending (group جمع)
- ع when it is not linked (section فرع)
```

Figure 6. Positional variants of Arabic characters

4.3.2. Phonetic keyboard input program

Certain languages, such as Chinese, are based on a vast number of ideograms, each one representing a particular concept. Recently (1980), simplified and varied versions of Chinese have been adopted by the People's Republic of China, on one side, and by Taiwan and Hong Kong, on the other. Several different keyboard input methods exist, such as input via character codes (e.g. **GB-2312-80** code) or the **Pinyin** method, which consists of a phonetic representation of ideograms in Roman characters (420 syllables, complemented by one of five tones per syllable).

MtScript.2.0 version includes a Chinese keyboard input program for GB-2312-80 and BIG-5 codes based on several input methods (Pinyin phonetic transcriptions, radicals, 4-corners, etc.).



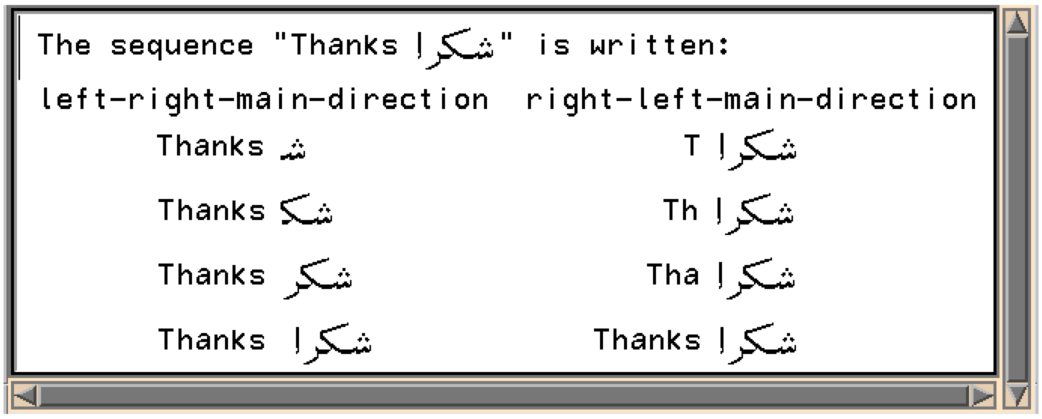
```
Chinese ideograms:
MtScript.2.0 includes programs to input Chinese ideograms.
One of the input methods uses Pinyin as follows:

多 ba1 文字条体
  芭 捌 扒 叭 吧 爸 八 疤 巴 岜 粑
Arabic and Hebrew texts: برنامج معالجة النصوص المتعددة اللغات .
```

Figure 7. Chinese ideograms keyboard input using Pinyin

4.4. Display and retrieval

As was stated earlier, a signifiant problem with displaying multilingual texts is the co-existence of opposite writing directions on the same line of text. Insertion and deletion of characters must take into account their writing direction, according to quite complex rules. **MtScript** allows the user to define interactively a main and a secondary writing direction for a text-zone (paragraph). The cursor moves only in the direction specified as the main one. When a sequence of characters is entered in a language written in the secondary direction, the cursor stays put and the characters are written in an insertion mode (see figure 7).



```
The sequence "Thanks شكرا" is written:
left-right-main-direction  right-left-main-direction
        Thanks ش                        T شكرا
        Thanks شك                       Th شكرا
        Thanks شكر                      Tha شكرا
        Thanks شكرا                     Thanks شكرا
```

Figure 8.   Bidirectional texts

Concerning the justification of multilingual texts including opposite writing directions, **MtScript version 1.1** allows multilingual texts to be only left or right justified or centred. However the left and right justification of multilingual texts is still under development. The justification of right-to-left written texts or bidirectional multilingual texts is a major problem. Contrary to Latin alphabet based texts which can be justified by using extra spaces, the justification of right-to-left written texts requires the use of **dynamic fonts**, where most of the characters can be contracted or stretched dynamically, according to the line length. In the example of Figure.9, the last character “Ba“ of the word “KaTaBa“ (to write) is drawn differently according to the end of the line. A special character such as “_“ can also be used to extend the character drawings.

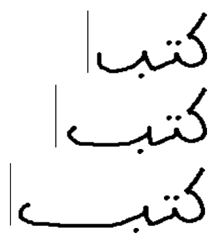


Figure 9. Dynamic font for Arabic text justification

Upcoming versions of **MtScript** will include mechanisms for multilingual text justification. To do that we currently are studing some significant works to define methods for justifying bidirectional texts including Hebrew or Arabic associated with other languages. That works have been proposed particularly by D.M.Berry [BERRY89, BERRY90, BERRY92], D.E.Knuth and P.MacKay [KNMC87].

4.5. Text import/export

In the current **MtScript.1.1** version multilingual texts are coded in the ISO-8859-*, gb2312-80, big5-0, jisx0208-1983-0 and ksc5601-1987-0 character standards[3] and each text is associated with a style file. This file contains relevant attributes of the text pieces, such as the language attribute, font, etc. Thus **MtScript** is able to edit imported multilingual texts coded in one or more of the above mentioned standards. If the text contains more than one language, then information must be provided to indicate the appropriate language for each piece of the text. From the opposite perspective, the multilingual texts produced by **MtScript** can be exported (even by Internet protocols: MIME, etc.) to other multilingual editors supporting the same coding standards. But as the style files cannot be interpreted by other editors, information must be provided in some other way to indicate the appropriate attributes for each of the pieces of the texts (particularly the language attribute).

5. Assessment and future developments

MtScript.1.1 has been distributed in compiled form[4] for Solaris/Linux systems via the Internet since May 1996 and it has been downloaded and tested by hundreds of users in more than 32 countries (in research centres and companies, etc.). A lot of users found **MtScript** easy to use (WYSIWYG) and compatible with standard character codes and standard transliteration rules. Some users have adopted **MtScript** to type new multilingual texts or to edit existing ones. Other users proposed to contribute to the development of **MtScript** by developing writing and transliteration rules for specific languages[5]. Certain companies and research centres working on software design have asked us to link **MtScript** with their applications (e.g., it has been linked to a machine translation system for editing Unix shell commands, etc.).

However the current version of **MtScript** presents several limits on which we are currently working. Each text is associated with a style file that is usable only by **MtScript**, which limits the text import/export abilities of the software even though the characters are represented by standard codes. SGML and TEI offer alternative ways of encoding multilingual texts, and are to a large extent meant for interchange [IDVE95]. SGML encoding would also have the benefit of not forcing the transfer of both the text file and its style file, without which the text is essentially meaningless.

[3] Expect for Arabic texts, but this has been fixed in the **MtScript.2.0** version currently under development.

[4] It is not excluded that next distributions of MtScript will be done via the GNU General Public License (source code freely available expect for commercial purposes).

[5] The distributed MtScript.1.1 version does not include the writing and the transliteration rules for all the included languages. Despite the fact that the formalisms are not yet easy to manipulate, users can define writing and transliteration rules for their specific languages.

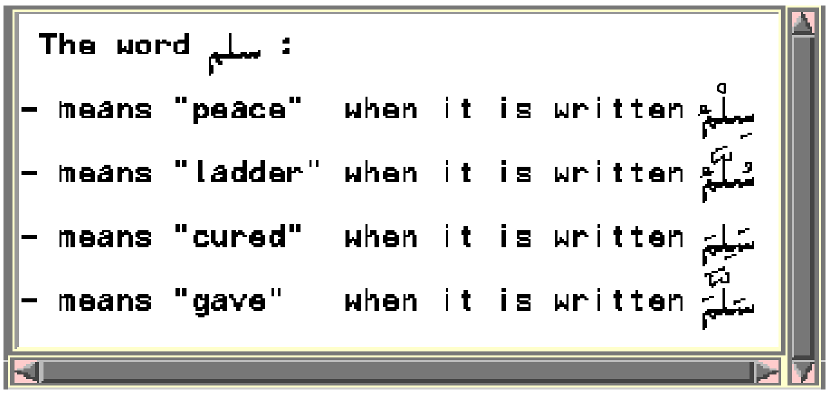
MtScript.2.0 is almost ready for distribution. This version includes new features, such as:

- Chinese keyboard input program (for GB-2312-80 and BIG-5 codes),
- ability to link the **MtScript** editor with external Unix commands (Unix shell, etc.),
- ability to use a Unix Spellchecker with many languages (ispell, etc.),
- ability to print Latin-alphabet-based texts (free PostScript fonts available),
- inclusion of other new languages,
- SGML text format (this module is developed but not fully integrated).

Other features at an earlier stage of development include:

- ISO 10646/UNICODE character coding,
- SGML and TEI document import/export,
- printing non-Latin languages (free PostScript fonts not available yet),
- HTML text encoding to link **MtScript** with a WWW browser,
- MIME data transfer to link **MtScript** with an electronic mail program,
- left and right justification of bi-directional texts.

Moreover we are planning to include vowels in the Arabic language, which are represented by diacritics written above or below consonants. Even though the Arabic texts one sees in newspapers and magazines do not generally contain vowels (the human reader can normally ascertain the meaning of vowel-less words given their context, thanks to the morpho-graphematical structure of the Arabic language), these are of prime importance in Arabic text-processing (in particular when compiling lexicons).



```
The word سلم :
- means "peace"   when it is written سِلْمُ
- means "ladder"  when it is written سُلَّمُ
- means "cured"   when it is written سَلِمَ
- means "gave"    when it is written سَلَّمَ
```

Figure 10. Arabic vowels

6. Conclusion

As the exchange of information among parties who speak different languages becomes more and more frequent, the multilingual text editor has become a fundamental tool for bringing about the globalization of information resources. **MtScript** is being developed with a view to answering the needs for coding, processing, and viewing tools required for multilingual documents in both European and non-European languages. This tool allows inputting, editing and storing multilingual texts, and can be linked to many applications requiring multilingual text processing and editing, such as the segmentation and morphological analysis of texts, machine-aided translation, multilingual dictionary design, and the localization of software and associated documents into different languages.

Aknowledgments

This work has been financied by the European Community in the framework of the MULTEXT Project. Several persons contributed to the improvment of **MtScript.** We acknowledge particularly Jean Véronis (coordinator of the MULTEXT project) for his help in designing **MtScript** and reading this paper, Nancy Ide (Vassar college, USA) for her help and her instructive comments, Emmanuel Flachaire (LPL) for the compilation of the Linux/Intel version, Greg Priest-Dorman (Vassar college) for intensive tests of **MtScript**, and Mark Leisher (CRL Laboratory) for his comments and his help with considerable resources. Moreover we acknowledge our colleagues for their help in designing **MtScript** and in reading this paper.

Bibliography

[ACC96] **http://www.accentsoft.com** (Accent Software International Ltd., Multilingual technologies on Windows).

[BENT91] P.M. Benton, "The Multilingual Edge", *BYTE*, March 1991, 124-132.

[BERRY89] Z.Becker, D.Berry, "triroff, an adaptation of the device-independant troff for formatting tri-directional text", *ELECTRONIC PUBLISHING*, Vol.2(3), October 1989, 119-142.

[BERRY90] U.Habusha, D.Berry, "vi.iv, a bi-directional version of the vi full-screen editor", *ELECTRONIC PUBLISHING*, Vol.3(2), May 1990, 65-91.

[BERRY92] J.Srouji, D.Berry, "Arabic formatting with ditroff/ffortid", *ELECTRONIC PUBLISHING*, Vol.5(4), December 1992, 163-208.

[BOUA90] A.M. Boualem, "The multilingual terminal", research report, INRIA Sophia Antipolis, January 1990, 1-4.

[BOUA93] A.M. Boualem, "ML-TASC: Système de traduction automatique multilingue dans un environnement à syntaxe contrôlée", SS'93, *7th annual High Performance Computing Conference*, Alberta, Canada, June 1993, 537-544

[BOUA95a] A.M. Boualem, "Multilingual text editing", *SNLP'95, The 2nd Symposium on Natural Language Processing*, NECTEC, C&C, Bangkok, August 1995, 336-342.

[BOUA95b] A.M. Boualem, "Arabic Language Processing", *SNLP'95, The 2nd Symposium on Natural Language Processing*, NECTEC, C&C, Bangkok, August 1995, 95-102.

[CBKH95] C. Bigelow, K. Holmes, “The design of a UNICODE font”, version française dans le *Cahier GUTenberg* n°20, May 1995, 81-102.

[CRL96] **http://crl.nmsu.edu** (Computing Research Laboratory at New Mexico State University, Research and software development in advanced computing applications: Natural language processing, artificial intelligence, graphical user interface design, etc.).

[IDVE95] N. Ide, J. Véronis, *The Text Encoding Intiative: Background and Context*, Kluwer Academic Publishers, Dordrecht, 1995.

[JAMG95] J. André, M. Goossens, "Codage des caractères et multi-linguisme: de l'ASCII à UNICODE et ISO/IEC-10646", *Cahier GUTenberg* n°20, May 1995, 1-54.

[KNMC87] D.E.Knuth, P.MacKay, "Mixing Right-to-left Texts with Left-to-right Texts", *TUGBoat*, 8(1), 1987, 14-25.

[LABO95] A. Labonté, "Input methods to enter characters from the repertoire of ISO/IEC 10646 with a keyboard or other input devices". *ISO/CEI JTC1/SC18/GT9 Working Draft*, February 1995. **ftp://ftp.funet.fi/pub/doc/charsets/ucs-input-methods**

[LANG93] *Language Coding Using ISO/IEC 6429*. Draft circulated in January 1993 by the European Standardization Organization CEN technical committee TC304. Available electronically at: **http://www.stonehand.com/unicode/standard/tc304.html/**

[MUL96] **http://www.lpl.univ-aix.fr/projects/multext/** (European Multext project: Multilingual Text Tools and Corpora, Developing standards, specifications and tools for the encoding and processing of linguistic corpora and resources for a wide variety of languages).

[YERG95] F. Yergeau, G. Nicol, G. Adams, M. Duerst, "Internationalization of the Hypertext Markup Language". *Internet Draft draft-ietf-html-i18n-02*, November 1995. **http://www.ics.uci.edu/pub/html/draft-ietf-html-i18n-02.txt**

[YHJP95] Y. Haralambous, J. Plaice, "W, une extension de TEX incluant UNICODE et des filtres de type Lex", *Cahier GUTenberg* n°20, May 1995, 55-79.