

## A glance at CJK support with X<sub>Ǝ</sub>TeX and LuaTeX

Antoine Bossard

### Abstract

From a typesetting point of view, the Chinese and Japanese writing systems are peculiar in that their characters are concatenated without ever using spaces to separate them or the meaning units (i.e., “words” in our occidental linguistic terminology) they form. And this is also true for sentences: although they are usually separated with punctuation marks such as periods, spaces remain unused. Conventional typesetting approaches, TeX in our case, thus need to be revised in order to support the languages of the CJK group: Chinese, Japanese and, to a lesser extent, Korean. While more or less complete solutions to this issue can be found, in this article we give and pedagogically discuss a minimalistic implementation of CJK support with the Unicode-capable X<sub>Ǝ</sub>TeX and LuaTeX typesetting systems.

### 1 Introduction

The Chinese, Japanese and Korean writing systems are conventionally gathered under the CJK appellation. The Chinese writing system consists of the Chinese characters, which can be in simplified or traditional form, amongst other character variants [1]. The (modern) Japanese writing system is made of the Chinese characters and the kana characters. The Chinese and Japanese writing systems concatenate characters without ever separating them with spaces. The Korean writing system consists mainly of hangul characters, with in addition the Chinese characters, which are however rarely used nowadays. Although modern Korean separates words with spaces, traditionally, the Korean writing system does not (as an illustration, see for instance Sejong the Great’s 15<sup>th</sup> century manuscript *Hunminjeongeum*<sup>1</sup>).

Notwithstanding other critical issues such as fonts, by not relying on spaces between characters or words, the CJK scripts are a challenge to conventional typesetting solutions such as TeX. In fact, the algorithms for word-breaking, which conventionally occurs at spaces (plus hyphenation), become inapplicable.

On a side note, even though we consider hereinafter the CJK writing systems, this discussion can be extended to related scripts such as Tangut and Vietnam’s Chữ Nôm.

<sup>1</sup> King Sejong (世宗) introduced hangul in the *Hunminjeongeum* (訓民正音) manuscript (1443–1446).

In this paper, we provide a glance at CJK support with X<sub>Ǝ</sub>TeX and LuaTeX by giving a minimalistic implementation for these oriental scripts. This work is both a proof of concept and a pedagogical discussion on how to achieve CJK support as simply as possible with the aforementioned typesetting solutions. Both X<sub>Ǝ</sub>TeX and LuaTeX support Unicode, which enables us to focus on typesetting issues, leaving encoding and font considerations aside.

The rest of this paper is organised as follows. Technical discussion of the proposed implementation is conducted in Section 2. The state of the art and paper contribution are summarised in Section 3. Finally, this paper is concluded in Section 4.

## 2 A minimalistic implementation

We describe here the proposed minimalistic implementation of CJK support with X<sub>Ǝ</sub>TeX and LuaTeX step by step in a pedagogical manner: paragraph management (Step 1) is addressed in Section 2.1, Latin text mingling (Step 2) in Section 2.2, Latin text paragraph (Step 3) in Section 2.3, Korean text paragraph (Step 4) in Section 2.4 and sophisticated line-breaking (Step 5) in Section 2.5. “Latin text” here designates text written with the Latin alphabet, or similar; for instance English and French text.

A handful of TeX commands are used hereinafter without being detailed; see [4] for those that are not self-explanatory. The document preamble specifies nothing particular. The fontspec package [10] is loaded for facilitated font manipulation, and, as detailed in the rest of this section, since it is considered without loss of generality that the document consists of Chinese or Japanese paragraphs by default, the main font of the document is set accordingly (e.g., `\setmainfont{Noto Serif CJK JP}` [3]).

### 2.1 Paragraph management

A conventional approach to break long character sequences (i.e., Chinese or Japanese characters in our case) is to insert between each two glyphs a small amount of horizontal space so that TeX can split the sequence across multiple lines (see for instance [13]). Without such extra space, line breaks could still occur thanks to hyphenation, but this is not applicable in the CJK case. We rely on a “scanner” macro to transform a paragraph by interleaving space between its characters. In practice, according to the TeX terminology, this extra space will be a horizontal skip of 0pt width and ±1pt stretch.

The scanner macro is a recursive process that takes one token (e.g., a character) as single parameter and outputs it with on its right extra horizontal space. The recursion stops when the parameter token is the stop signal (more on this later); in this case, the macro outputs `\par`, thus materialising the end of the paragraph. The scanner macro `\cjk@scan` is defined as follows.

```
\def\cjk@scan#1{% #1: single token
\ifx#1\cjk@stop% stop signal detected
\par% so, complete the paragraph
\else
#1% display the current character
\hskip 0pt plus 1pt minus 1pt\relax% space
\expandafter\cjk@scan% recursive call
\fi
}
```

The above scanner is started by the `\cjk@scanstart` macro whose primary objective is to append the stop signal `\cjk@stop` at the end of the paragraph that is about to be transformed. This initial macro takes one parameter: the paragraph to transform. In a pattern matching fashion, a paragraph is taken as a whole by setting `\par` as delimiter for the parameter of the `\cjk@scanstart` macro. This will require inserting `\par` once the paragraph has been transformed though, since the `\par` command that ends the paragraph is treated as a delimiter by the macro and thus skipped. In addition, each paragraph needs to be ended by a blank line (i.e., `\par`) to content this pattern matching. The scanner starting macro is given below.

```
\def\cjk@scanstart#1\par{% #1: paragraph
\cjk@scan#1\cjk@stop% append \cjk@stop
}
```

In this work, paragraphs are considered to be written in Chinese or Japanese by default. Hence, paragraph typesetting mode selection by means of a command such as `\CHJP{text}` is not suitable. We rely on the `\everypar` token parameter so as to trigger the transformation of each paragraph with the scanner previously described. This is simply done as follows:

```
\everypar={\cjk@scanstart}
```

or, in a safer manner [2]:

```
\everypar=\expandafter{\the\everypar
\cjk@scanstart}
```

人民身體之自由應予保障。除現行犯之逮捕  
人民因犯罪嫌疑被逮捕拘禁時，其逮捕拘禁  
人民身體之自由應予保障。除現行犯之逮捕  
由法律另定外，非經司法或警察機關依法定程  
序，不得逮捕拘禁。非由法院依法定程序，不  
得審問處罰。非依法定程序之逮捕、拘禁、審  
問、處罰，得拒絕之。  
人民因犯罪嫌疑被逮捕拘禁時，其逮捕拘禁  
機關應將逮捕拘禁原因，以書面告知本人及其  
本人指定之親友，並至遲於二十四小時內移送  
該管法院審問。本人或他人亦得聲請該管法院  
，於二十四小時內向逮捕之機關提審。

(a)

(b)

**Figure 1:** Before (a) and after (b) paragraph transformation: line breaking now enabled (traditional Chinese text example).

An illustration of the result of this paragraph transformation is given in Figure 1 (Chinese and Japanese paragraphs).

## 2.2 Latin text mingling

It is often the case that Latin text such as English words, expressions or sentences is mingled within Chinese or Japanese paragraphs. In the previously described paragraph transformation method, spaces, if any, are “gobbled” and never passed as parameter for the scanner macro `\cjk@scan`. This is not really a problem for Chinese and Japanese text since as explained they do not rely on spaces. But now that we are considering Latin text mingling in such paragraphs, spaces need to be retained since Latin text, such as English, rely on spaces for instance to separate words.

Without going too much into details, to force  $\TeX$  to also pass spaces as parameters to the scanner macro, spaces need to be made *active*, as per the  $\TeX$  terminology. Hence, it suffices to call the `\obeyspaces` macro, whose purpose is exactly to make spaces active, at the beginning of the document. In addition, the scanner macro is refined to avoid adding extra space when the current character is a space; see below.

```
\def\cjk@scan#1{%
\ifx#1\cjk@stop
\par
\else
#1%
\if#1\space% no extra space if #1 is a space
\else
\hskip 0pt plus 1pt minus 1pt\relax
\fi
\expandafter\cjk@scan
\fi
}
```

Two remarks are made next. First, it should be noted that Latin text mingled within Chinese or Japanese paragraphs is treated just as Chinese

日本国民は、正当に選挙された国会における代表者を通じて行動し、われらとわれらの子孫のために、諸国民との協和による成果と、わが国全土にわたつて自由のもたらす恵沢を確保し、政府の行為によつて再び戦争の惨禍が起ることのないやうにすることを決意し、ここに主権が国民に存することを宣言し、この憲法を確定する。そもそも国政は、国民の厳粛な信託によるものであつて、その権威は国民に由来し、その権力は国民の代表者がこれを行使し、その福利は国民がこれを享受する。これは人類普遍の原理であり、この憲法は、かかる原理に基くものである。われらは、これに反する一切の憲法、法令及び詔勅を排除する。 We,theJapanesepeople...

(a)

日本国民は、正当に選挙された国会における代表者を通じて行動し、われらとわれらの子孫のために、諸国民との協和による成果と、わが国全土にわたつて自由のもたらす恵沢を確保し、政府の行為によつて再び戦争の惨禍が起ることのないやうにすることを決意し、ここに主権が国民に存することを宣言し、この憲法を確定する。そもそも国政は、国民の厳粛な信託によるものであつて、その権威は国民に由来し、その権力は国民の代表者がこれを行使し、その福利は国民がこれを享受する。これは人類普遍の原理であり、この憲法は、かかる原理に基くものである。われらは、これに反する一切の憲法、法令及び詔勅を排除する。 We,theJapanesepeople...

(b)

**Figure 2:** Before (a) and after (b) making spaces active: Latin text mingling now retains spaces (Japanese text example).

or Japanese text: extra space is inserted between glyphs. Therefore, line and word breaking for mingled Latin text can occur anywhere, and thus no word-breaking by hyphenation will ever happen since never necessary due to the extra space added between glyphs. Second, even though no extra space is added after a space character, extra space is still added before a space character. This issue will be tackled in a subsequent section.

An illustration of the result of this refined paragraph transformation is given in Figure 2.

### 2.3 Latin text paragraph

Because the `\obeyspaces` macro has been called so as to typeset Chinese and Japanese paragraphs, Latin text paragraphs would be typeset just as those, that is, with extra space added between consecutive glyphs (except after spaces). As a result, and as explained above, line and word breaking would not be satisfactory.

Hence, we next enable the proper typesetting of Latin text paragraphs, that is, paragraphs that include spaces between words. To this end, we define the `\iflatin` conditional statement that will be used to distinguish Latin text paragraphs from others. The flag command `\latinfalse` is called at the beginning of the document to reflect that Chinese and Japanese paragraphs are by default. Latin text paragraphs are marked as such by calling the flag command `\latintrue` at the beginning of the paragraph. The scanner starting macro `\cjk@scanstart` is adjusted so as to not start the scanner in case the Latin flag is set.

As the `\obeyspaces` macro has been previously called, spaces are active characters; this setting needs to be reverted in the case of Latin text paragraph in order to have proper line and word breaking. Hence, the scanner starting macro in addition reverts spaces from the active state back to their

日本国民は、正当に選挙された国会における代表者を通じて行動し、われらとわれらの子孫のために、諸国民との協和による成果と、わが国全土にわたつて自由のもたらす恵沢を確保し、政府の行為によつて再び戦争の惨禍が起ることのないやうにすることを決意し、ここに主権が国民に存することを宣言し、この憲法を確定する。

We, the Japanese people, acting through our duly elected representatives in the National Diet, determined that we shall secure for ourselves and our posterity the fruits of peaceful cooperation with all nations and the blessings of liberty throughout this land, and resolved that never again shall we be visited with the horrors of war through the action of government, do proclaim that sovereign power resides with the people and do firmly establish this Constitution.

(a)

日本国民は、正当に選挙された国会における代表者を通じて行動し、われらとわれらの子孫のために、諸国民との協和による成果と、わが国全土にわたつて自由のもたらす恵沢を確保し、政府の行為によつて再び戦争の惨禍が起ることのないやうにすることを決意し、ここに主権が国民に存することを宣言し、この憲法を確定する。

We, the Japanese people, acting through our duly elected representatives in the National Diet, determined that we shall secure for ourselves and our posterity the fruits of peaceful cooperation with all nations and the blessings of liberty throughout this land, and resolved that never again shall we be visited with the horrors of war through the action of government, do proclaim that sovereign power resides with the people and do firmly establish this Constitution.

(b)

**Figure 3:** Before (a) and after (b) Latin mode enabling: Latin text now properly typeset (Japanese and English text example).

default state in the case of a Latin text paragraph. The refined code is given below.

```
\newif\iflatin % flag to detect whether to scan
\latinfalse % flag initially set to false
```

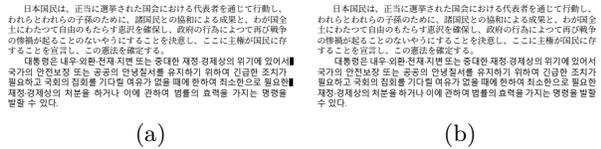
```
\def\cjk@scanstart#1\par{
\iflatin% if Latin text paragraph, don't scan
\catcode'\ =10% revert \obeyspaces
#1\par% display the paragraph normally
\latinfalse
\else
\cjk@scan#1\cjk@stop
\fi
}
```

An illustration of the result of this refined paragraph transformation is given in Figure 3.

### 2.4 Korean text paragraph

Let us now discuss the case of Korean text paragraph typesetting. As mentioned in introduction, modern Korean relies on spaces to separate words. Hence, such paragraphs are treated as a Latin text paragraph, concretely being marked with the `\latintrue` flag. Yet, because Korean glyphs (i.e., hangul or hanja) are wider than Latin ones, the width of spaces is adjusted. In addition, a font switch is also used to select a Korean font since it is frequent that Korean glyphs are not included inside the default font used for Chinese and Japanese paragraph typesetting.

Such settings are applied at the beginning of the paragraph, thus embedding the paragraph into a group for font selection and the adjusted space setting. Hence, the paragraph starts with



**Figure 4:** Before (a) and after (b) space width adjustment for Korean text: no more overfull hboxes (Japanese and Korean text example).

a ‘{’ token and it is required to leave the vertical mode for proper parsing of the paragraph when used as parameter of the scanner starting macro `\cjk@scanstart`. Precisely, the problem with starting the paragraph with a command like `{\malgun` (e.g., font switch) is that  $\text{\TeX}$  is still in vertical mode when it processes it. Switching to horizontal mode starts a new paragraph and thus triggers `\everypar`, but then with an unmatched ‘}’ remaining (i.e., the one corresponding to, say, the font switch) at the end of the paragraph, and thus the parsing error.

For convenience, these Korean text paragraph settings are gathered under the `\korean{}` macro as defined below.

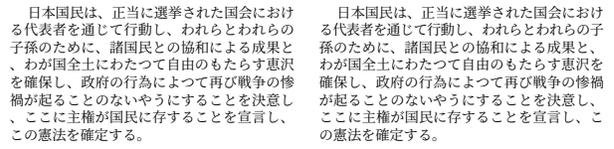
```
\def\korean#1{
  \latintrue% activate the Latin mode
  \leavevmode% leave the vertical mode
  {% Adjust the space size:
   \spaceskip=\fontdimen2\font plus
   3\fontdimen3\font minus
   3\fontdimen4\font% ×3 stretch and shrink
   \malgun #1% Korean font switch
  }
}
```

It should be noted that this redefinition of `\spaceskip` for the current paragraph would also be applied for Latin text mingled within a Korean paragraph. Furthermore, this font selection process – yet without activating the Latin mode and adjusting the space width – could also be used in the case where distinct fonts for Chinese and Japanese text are required.

An illustration of the result of this paragraph typesetting is given in Figure 4. Before space width adjustment, the overfull hboxes materialised by the two black boxes should be noticed.

### 2.5 Sophisticated line-breaking

Just as, say, in French, where line breaks are not allowed before the punctuation marks ‘:’, ‘;’, ‘!’ and so on – even though these need to be preceded by a space and are thus typical usages of non-breaking spaces – CJK typesetting expectedly forbids line



**Figure 5:** Paragraph transformation by the original (a) and new (b) scanner macro: no more line break before a comma (Japanese text example).

breaks before punctuation marks such as commas and periods.

We derive in this section a new scanner macro, `\cjk@scanbis`, to tackle this remaining problem. The approach is simple: refrain from adding extra space after the current character when the next one is a punctuation mark. And at the same time, this new scanner allows us to solve the aforementioned incongruity of extra space being added before a space character in Latin text paragraphs.

In practice, the new scanner takes two tokens as parameters instead of one: the first parameter is the currently processed token and the second one is the next token in line. The recursive call is also updated since now expecting two tokens as parameters instead of one; see below.

```
\def\cjk@scanbis#1#2{% two tokens passed
  #1%
  \ifx#2\cjk@stop
  \par
  \else
  \if#2,% no extra space before character ‘,’
  \else\if#2。% idem before character ‘。’
  \else\if#2\space% idem before a space
  \else\if#1\space% idem after a space
  \else\hskip 0pt plus 1pt minus 1pt\relax
  \fi\fi\fi\fi
  \expandafter\cjk@scanbis\expandafter#2%
  \fi
}
```

Note that similar additional conditions for other CJK punctuation marks can be appended if needed. Besides, in the scanner macro `\cjk@scanstart`, the expression `\cjk@scan#1\cjk@stop` is naturally changed to `\cjk@scanbis#1\cjk@stop`.

An illustration of the effect of this new scanner is shown in Figure 5.

### 3 State of the art and contribution

Early solutions for CJK support within the  $\text{\TeX}$  ecosystem include the CJK package [5] and the

Japanese  $\TeX$  system  $p\TeX$  [7]. While the former provides some support for Unicode, the latter does not. Notably, the CJK package has partial support for vertical typesetting. Regarding Korean, the `hlatex` package [12] enables the processing by  $\LaTeX$  of KS X 1001 encoded files, and of UTF-8 files via the obsolete  $\TeX$  extension Omega [9]. Omega also has some support for multi-directional CJK typesetting.

More recent alternatives include the `xeCJK` package [6], which is dedicated to  $X_{\text{CJK}}\TeX$  (i.e., no  $\text{Lua}\TeX$  support). This package is very large as it consists of more than 14,000 lines of macro code. As of 2018, it is only documented in Chinese. Another extensive package, `LuaTeX-ja` [11], is available, this time restricted to the support for Japanese with  $\text{Lua}\TeX$ . Finally, `upLaTeX` [8], another system dedicated to Japanese can also be cited; it is based on  $p\LaTeX$ , which is in turn based on  $p\TeX$ .

Even if the above are more or less complete solutions to the CJK typesetting issue with  $\TeX$ , we have presented in this paper a very simple solution, which neither requires a separate  $\TeX$  system such as  $p\TeX$  nor advanced  $\TeX$  capacities such as `xtemplate`, `LaTeX3`, etc., unlike, for instance, `xeCJK`. With only a few lines of macro code, we have described how to add basic yet arguably competent support for CJK to both  $X_{\text{CJK}}\TeX$  and  $\text{Lua}\TeX$ , indistinctly. The  $X_{\text{CJK}}\TeX$ ,  $\text{Lua}\TeX$  flexibility has been retained: no extra layer has been piled as, for instance, with `xeCJK` (e.g., the `\setCJKmainfont` command). Moreover, the complexity induced by packages such as `xeCJK` is likely to be a threat to the compatibility with other packages, as well as with online compilation systems such as those employed by scientific publishers.

#### 4 Conclusions

It is well known that the Chinese, Japanese and Korean writing systems are challenging for the typesetting solutions such as  $\TeX$  that were originally designed for Latin text. Various extensions and packages were proposed to support CJK in  $\TeX$ , with uneven success. Such solutions are in most cases, if not all, extensive – not to say invasive – additions to the  $\TeX$  ecosystem. In this paper, relying on the Unicode-capable  $X_{\text{CJK}}\TeX$  and  $\text{Lua}\TeX$  systems, we have presented and pedagogically discussed a minimalistic solution to this CJK typesetting issue. With only a few lines of macro code, we have shown that satisfactory CJK support can be achieved: paragraph management, Latin text mingling and sophisticated line-breaking are examples of the addressed typesetting issues.

Regarding future works, given its still rather frequent usage, right-to-left horizontal typesetting would be a useful addition to this pedagogical discussion on CJK typesetting. Furthermore, even though a complex issue for  $\TeX$ , right-to-left vertical typesetting is a meaningful objective as it is ubiquitous for the CJK writing systems.

#### Acknowledgements

The author is thankful towards Keiichi Kaneko (Tokyo University of Agriculture and Technology, Japan) and Takeyuki Nagao (Chiba University of Commerce, Japan) for their insightful advices.

#### References

- [1] A. Bossard. *Chinese Characters, Deciphered*. Kanagawa University Press, Yokohama, Japan, 2018.
- [2] S. Checkoway. *The everyhook package*, 11 2014. Package documentation. <https://ctan.org/pkg/everyhook> (last accessed November 2018).
- [3] Google. Google Noto fonts, 2017. <https://google.com/get/noto> (last accessed December 2018).
- [4] D. E. Knuth. *The TeXbook*. Addison-Wesley, Boston, MA, USA, 1986.
- [5] W. Lemberg. *CJK*, 4 2015. Package documentation. <https://ctan.org/pkg/cjk> (last accessed November 2018).
- [6] L. Liu and Q. Lee. *xeCJK 宏包 (in Chinese)*, 4 2018. Package documentation. <https://ctan.org/pkg/xeCJK> (last accessed November 2018).
- [7] K. Nakano, Japanese  $\TeX$  Development Community, and TTK. *About pLaTeX 2 $\epsilon$* , 9 2018. Package documentation. <https://ctan.org/pkg/platex> (last accessed November 2018).
- [8] K. Nakano, Japanese  $\TeX$  Development Community, and TTK. *About upLaTeX 2 $\epsilon$* , 4 2018. Package documentation. <https://ctan.org/pkg/uplax> (last accessed November 2018).
- [9] J. Plaice and Y. Haralambous. The latest developments in  $\Omega$ . *TUGboat* 17(2):181–183, 6 1996.
- [10] W. Robertson. *The fontspec package – Font selection for X $\text{La}\TeX$  and Lua $\text{La}\TeX$* , 7 2018. Package documentation. <https://ctan.org/pkg/fontspec> (last accessed December 2018).
- [11] The  $\text{Lua}\TeX$ -ja project team. *The LuaTeX-ja package*, 11 2018. Package documentation. <https://ctan.org/pkg/luatexja> (last accessed November 2018).
- [12] K. Un. 한글라텍 길잡이 (in Korean), 4 2005. Package documentation. <https://ctan.org/pkg/hlatex> (last accessed December 2018).

- [13] B. Veytsman. *Splitting Long Sequences of Letters (DNA, RNA, Proteins, Etc.)*, 8 2006. Package documentation. <https://ctan.org/pkg/seqsplit> (last accessed November 2018).

## Appendix

The placeholder text used in the various illustrations of this article is in the public domain as detailed below. Figure 1: the placeholder text is the two first paragraphs of Article 8 of the Chinese constitution (1947), written with traditional Chinese. Figure 2: the placeholder text is the first paragraph of the Japanese constitution (1946), followed by the first few words of the corresponding official English translation. Figure 3: the placeholder text is the first sentence of the first paragraph of the Japanese constitution (1946), followed by the corresponding official English translation. Figure 4: the placeholder text is the first sentence of the first paragraph of the Japanese constitution (1946), followed by the first paragraph of Article 76 of the South Korean constitution (1988). Figure 5: the placeholder text is the first sentence of the first paragraph of the Japanese constitution (1946).

◇ Antoine Bossard  
Graduate School of Science  
Kanagawa University  
2946 Tsuchiya, Hiratsuka,  
Kanagawa 259-1293, Japan  
[abossard@kanagawa-u.ac.jp](mailto:abossard@kanagawa-u.ac.jp)