LaTeX: rendering texts in Japanese and Chinese (on Linux, Windows, and other OSes)

Pierre S. Caboche (kabocha piēru 南瓜石) *

July 11, 2022

Abstract

This article tries to solve some common problems in LaTeX when dealing with texts in Chinese, Japanese, and other languages that use a non-Latin writing system.

The first problem we intend to solve is related to *portability*: making sure that a LaTeX document containing texts in Japanese, Chinese (or other languages) can still be rendered in Windows®, Linux, or other systems (i.e. all our dependencies will have to be freely available for different OSes).

This is a prerequisite to the second problem we want to solve: adding *ruby* characters, (e.g. *furigana*, *pinyin*) to texts in Japanese, Chinese (and other languages).

However, this article needed to go beyond solving those two issues; and for a better understanding we also had to provide some background information about related subjects, including: the type of fonts we'll use (and where to find them), LaTeX, *ruby* characters, and even a few words about the Japanese and Chinese language.

By the end of this article, we will see what it involves to add *ruby* characters in *LibreOffice* and *Microsoft® Word®*, and see how they compare to LaTeX for this type of task.

*The name Caboche (meaning "head" in French) sounds similar to the Japanese word for "pumpkin" ("kabocha" - 南瓜), while the given name Pierre (ピエール, *piēru*) means "stone" in French, so I use the character for "stone" (石 - *ishi*).

Background

I started to use LaTeX to write documents containing a lot of Japanese text and *furigana*[1]. From my experience (and by using some of the techniques described in this article), adding *furigana* was considerably faster to do in LaTeX than in either *Word*® or *LibreOffice*, as we'll discover towards the end of this article…

However, when I switched to Linux, I discovered that my LaTeX documents didn't render at all.

When I tried to look for a solution online, I found a lot of documents whose advice were either:

- *outdated*, as they relied on the obsolete packages
- *not portable*: they were written with Windows® in mind, and recommended the use of fonts that are not readily available on other systems (e.g. *Meiryo*)

I eventually found a solution to those issues, I decided to share my findings in this article.

Goal

Our goal in this article is to learn how to perform the following:

- in LaTeX, display texts in Chinese, Japanese, etc.
  …without relying on proprietary fonts, which might not be available on Linux
- add *ruby* characters, especially Japanese *furigana* (e.g. 南瓜(カボチャ)) and Chinese *pinyin* (e.g. 南瓜(nán guā))

Methodology

To achieve our goals, we will do the following:

- install the *Noto Fonts* for the relevant languages
- install LaTeX
- render documents containing texts in Chinese, Japanese, Korean, etc.
  …in a way that works on Windows, Linux, and other systems
- add *furigana* to text in Japanese with the `ruby` package, as well as a *custom macro*
- add *pinyin* to text in Mandarin Chinese with the `xpinyin` package
- perform the same tasks in *LibreOffice* (mini-guide included), and compare it with our solution in LaTeX

The rest of this article goes into more details about *what* those tools are, *how* to use them, and *why*.

[1] one of my hobbies is to study the lyrics of the Japanese songs I like, then try to sing them at the *karaoke*. LaTeX allows me to quickly add *furigana* to the lyrics, or any other Japanese text

What are *ruby* and *furigana*?

Ruby characters are annotations usually placed on top of[2] Chinese, Japanese, or Korean characters[3], which are usually used to show the pronunciation of such characters[4].

When adding *ruby* characters to texts in Standard Mandarin Chinese, *pinyin* (see below) are usually used as *ruby*.

Below is an example of *pinyin* used as *ruby*:

xuě huā piāo piāo　běi fēng xiāo xiāo
雪花飄飄 北風蕭蕭
tiān de　yī　piàn cāng máng
天地 一片蒼茫

In Japanese, *ruby* characters are usually called *furigana*.

Below is the word "*furigana*" (振り仮名), with *furigana* added to it:

ふ　が　な
振り仮名
fu　ri　ga　na
振り仮名

Ruby characters may also be referred to as *rubi*, and may be plurialised as: *ruby*, *rubi*, or *rubies* (I tend to use the phrase "*ruby* characters" to avoid the confusion between singular and plural).

What is *pinyin*?

pinyin is the official romanization system for Standard Mandarin Chinese.

The name "*pinyin*" comes from *"Hànyǔ Pīnyīn"* (汉语拼音), literally: *"to spell the sound of the Han language"*.

pinyin can be used on their own (e.g. "hàn yǔ pīn yīn") or as *ruby* characters (e.g. hàn yǔ pīn yīn
汉语拼音).

About foreign loanwords

Words of Chinese or Japanese origins are invariable in English. For example, the plural of *anime* or *manga* is "*anime*", "*manga*".

As such, the word "*kanji*" may refer to either one *kanji* (i.e. Chinese character, also used in Japanese) or several *kanji*.

Foreign loanwords (therefore, invariable in plural) used in this article include: *kanji*, *hiragana*, *katakana*, *furigana*, *pinyin*.

[2] or to the right, if the text is displayed vertically
[3] *ruby* characters can technically be used in other languages too
[4] *ruby* characters have other usages, but are mainly used to indicate pronunciation

Conventions

Some commands in this article need to be executed with *administrator* privileges (on Linux, that means *root*) to perform operations such as: installing new software on the system, modifying some system configuration, etc.

If you have *administrator* privileges on your machine, please read on...

> In this article, we indicate that a Linux command needs to be executed with *root* privileges by using:
>
> `sudo`
>
> However, please note that there are cases when the `sudo` command will not work; for example, if the current user does not appear in the list of `sudoers` (file `/etc/sudoers`).
>
> If that is your case, please use whichever method you normally use to run a command as *root* (e.g. *doas*, *su*), while keeping in mind that being logged in as *root* is extremely bad practice.

If you do NOT have *administrator* privileges on your machine, please contact your administrator.

General Information

This document was first published at:

https://pcaboche.github.io

Legal

Last revision: 11 July 2022

- Singular terms shall include the plural forms and vice versa.
- this Document is Copyright 2022 Pierre Caboche. All rights reserved. No unauthorised distribution allowed.
- the LaTeX Code which produced this Document is Copyright 2022 Pierre Caboche. All rights reserved.
- this Document also features Source Code (also known as Code Snippet), subject to different licenses.
  - if no license is mentionned, it is to be assumed that the Source Code is *proprietary*, and the property of its author and copyright holder.
  - if a license is mentionned (in a comment or through other means), you need to refer to the full description of the license.

    Below is an example of comment, indicating that the featured Source Code is subject to the BSD License, with mention of the copyright year and copyright holder:

```
# This source code is under the BSD License
# Copyright 2022 Pierre S. Caboche
```

- all of the Content (including Document, LaTeX Code, Source Code) IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ”AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
- the Copyright Holders reserve the right to amend this Legal Notice at any time, without prior notification. The latest version of the Legal Notice supersedes and replaces all prior versions of it.

BSD license

Wherever the BSD license is mentionned, the following applies:

Contents

I Fonts **8**

II LaTeX **15**

III Japanese texts in LaTeX **20**

IV Chinese texts in LaTeX **25**

V LaTeX vs. *LibreOffice*, *Word*®… **28**

VI Conclusion **30**

Part I
Fonts

In order to properly render LaTeX documents containing CJK characters (Chinese, Japanese, Korean), you will need to use fonts capable of displaying such characters[5].
Developing fonts that support CJK characters is very costly, and therefore such fonts may not be readily available across all Operating Systems.

For example, when switching to Linux, I didn't have access to the *Meiryo* font anymore because it is © *Microsoft Corporation*. So I had to find some portable alternative...
The Noto font family (Google, 2022) were designed to solve this problem.

The Noto fonts are free (under the "SIL Open Font License") and were commissioned by Google. Their goal is to cover a wide range of languages and writing scripts (including Chinese, Japanese, and Korean). Not all CJK ideographs are covered (30,000 of the nearly 75,000 CJK unified ideographs), but the most commonly used characters seem to be represented. (Wikipedia, 2022c)

If you were to install all the Noto fonts in their entirety (i.e. for every language and script available), this would end up occupying more than 1.1 GB of disk space (as of time of writing). To reduce space, it is recommended to install only the fonts that you intend to use.

Etymology

> The name "Noto" stands for "**no** more **to**fu". When a character cannot be rendered by a computer program, some of these programs (e.g. web browsers) show a substitute character instead (usually in the form of a small rectangle). (Wikipedia, 2022c)

Those characters are sometimes colloquially referred to as "tofu", due to their resemblance with a block of tofu. Also, such substitute characters were quite likely to appear when trying to render texts from languages in regions like China, Japan, Korea...[6]

The goal of "Noto" is to eliminate those "tofu" characters (by properly rendering texts that use different writing systems).

Portability

By switching to the Noto fonts, your documents will look different (when compared to using the proprietary fonts available by default on some Operating Systems) but you will gain in portability.
Noto fonts are available at: `https://www.google.com/get/noto/`

[5] the same is true for other writing systems, each requiring specialised fonts.

[6] if the slang had developed in another part of the world, then "tofu" characters might be have known by some other name (and probably some other food-related item too). It isn't hard to imagine that in some parallel universe, these might be referred to as "paneer" characters instead...

Later we'll see how to easily install the Noto fonts on Linux, but first we'll need to determine which fonts are available, and which packages we'll need.

Noto fonts packages

Using the package manager for your linux distribution (e.g. `dnf`, `apt`, `pacman`, etc.), we'll look for the packages containing the word **noto**.

The list of matching results is very long (covering a large list of languages and scripts), so we'll filter even further (looking for terms like "cjk" or "japanese").

For example, in Fedora:

```
$ dnf search noto | grep -i -E 'cjk|japanese'
```

Searching for just "japanese" will give us the following:

```
$ dnf search noto | grep -i 'japanese'
google-noto-sans-cjk-jp-fonts.noarch : Japanese Multilingual Sans OTF
    ↪ font files for google-noto-cjk-fonts
google-noto-sans-jp-fonts.noarch : Japanese Region-specific Sans OTF font
    ↪  files for google-noto-cjk-fonts
google-noto-sans-mono-cjk-jp-fonts.noarch : Japanese Multilingual Sans
    ↪ Mono OTF font files for google-noto-cjk-fonts
google-noto-serif-cjk-jp-fonts.noarch : Japanese Multilingual Serif OTF
    ↪ font files for google-noto-cjk-fonts
google-noto-serif-jp-fonts.noarch : Japanese Region-specific Serif OTF
    ↪ font files for google-noto-cjk-fonts
```

Here is some explanation regarding the package names...

Typefaces

When talking about CJK fonts, the terms Serif and Sans-serif have the following meaning:

Serif
: (roman)
means that the font will show the brush strokes

Sans serif
: (sans, sans-serif, gothic)
means that brush strokes are not present

To better illustrate the difference, we'll create a LaTeX document containing a few examples (at the moment we'll focus mainly on the document output. In later chapters we'll see the details of how the document works):

example-01-typefaces.tex

```
% This source code is under the BSD License
% Copyright 2022 Pierre S. Caboche

% Note: use XeLaTeX for rendering
%!TEX encoding = UTF-8

\documentclass{article}

\usepackage{xeCJK}
\setCJKmainfont{Noto Serif JP}
\setCJKsansfont{Noto Sans JP}
\setCJKmonofont{Noto Sans Mono CJK JP}

\begin{document}
  \begin{itemize}
    \item
      \textrm{
        Serif \emph{(Roman)}:
        \input{"sample-text.tex"}
      }
    \item
      \textsf{
        Sans Serif \emph{(Gothic)}:
        \input{"sample-text.tex"}
      }
    \item
      \texttt{
        Typewriter \emph{(monospace)}:
        \input{"sample-text.tex"}
      }
  \end{itemize}
\end{document}
```

sample-text.tex

```
% This source code is under the BSD License
% Copyright 2022 Pierre S. Caboche

\begin{itemize}
  \item regular
  \item \emph{emphasis} (usually in italics)
  \item \textit{italic}
  \item \textsc{small caps}
  \item 日本語 (\emph{kanji}, Regular)
  \item \textbf{日本語} (\emph{kanji}, Bold)
  \item \textit{日本語} (\emph{kanji} can't be italicized)
  \item \textsc{日本語} (\emph{kanji} don't have lowercase/uppercase\dots
    ↪ )
  \item にほんご (\emph{hiragana})
  \item ニホンゴ (\emph{katakana})
  \item ﾆﾎﾝｺﾞ (half-width \emph{katakana})
\end{itemize}
```

This shows the differences between those typefaces...

Differences between *Serif*, *Sans-Serif*, and *Monospace*

- Serif *(Roman)*:
  - regular
  - *emphasis* (usually in italics)
  - *italic*
  - small caps
  - 日本語 (*kanji*, Regular)
  - **日本語** (*kanji*, Bold)
  - 日本語 (*kanji* can't be italicized)
  - 日本語 (*kanji* don't have lowercase/uppercase…)
  - にほんご (*hiragana*)
  - ニホンゴ (*katakana*)
  - ﾆﾎﾝｺﾞ (half-width *katakana*)
- Sans Serif *(Gothic)*:
  - regular
  - *emphasis* (usually in italics)
  - *italic*
  - small caps
  - 日本語 (*kanji*, Regular)
  - **日本語** (*kanji*, Bold)
  - 日本語 (*kanji* can't be italicized)
  - 日本語 (*kanji* don't have lowercase/uppercase…)
  - にほんご (*hiragana*)
  - ニホンゴ (*katakana*)
  - ﾆﾎﾝｺﾞ (half-width *katakana*)
- `Typewriter` *(monospace)*:
  - `regular`
  - *`emphasis`* `(usually in italics)`
  - *`italic`*
  - `SMALL CAPS`
  - 日本語 (*`kanji`*`, Regular)`
  - **日本語** (*`kanji`*`, Bold)`
  - 日本語 (*`kanji`* `can't be italicized)`
  - 日本語 (*`kanji`* `don't have lowercase/uppercase…)`
  - にほんご (*`hiragana`*`)`
  - ニホンゴ (*`katakana`*`)`
  - ﾆﾎﾝｺﾞ (`half-width` *`katakana`*`)`

From the previous example, we can see that:

- a font with *Serif* will show the brush strokes, and therefore will give a better idea of how a *kanji* is to be drawn
- a *Sans-serif* font, on the other hand, tends to be easier to read

Characters in Japanese, Chinese, and Korean are of fixed width (monospaced).

Japanese also has *half-width kana*, i.e. *katakana* characters which are half the width of regular *kana*, and are used only in certain context where display is limited in size. (Wikipedia, 2022a)

Characters in Japanese, Chinese, and Korean cannot be put in italics, and are not subject to “casing” (i.e. there is no distinction between lowercase and uppercase).

Writing systems

Earlier on, we saw that Fedora Linux provided packages with names like:

```
google-noto-sans-cjk-jp-fonts.noarch
google-noto-sans-jp-fonts.noarch
google-noto-sans-mono-cjk-jp-fonts.noarch
google-noto-serif-cjk-jp-fonts.noarch
google-noto-serif-jp-fonts.noarch
...
```

We already know that:

google-noto- … -fonts
: represent the *Google Noto* font families

sans-, serif-, mono-
: are the different typefaces available

The remaining part of the package name corresponds to the font target (in terms of language, script, use, etc.)

Below are some examples:

jp
: Japanese

kr
: Korean

sc
: Simplified Chinese

tc
: Traditional Chinese

hk
: Traditional Chinese Region-specific

cjk-jp*, *cjk-kr*, *cjk-sc*, *cjk-tc*, *cjk-hk
: Multilingual (Chinese, Japanese, Korean) versions of the above

myanmar
: Myanmar

myanmar-ui
: Myanmar UI font (i.e. targeted towards apps and software user interfaces)

myanmar-vf
: Myanmar variable font

myanmar-vf-ui
: Myanmar UI variable font

…

Installation

After going through the list of available packages, we need to choose the ones we'll need and install them.

Windows

For Windows, you need to:

- go to the Noto Fonts website (`https://www.google.com/get/noto/`)
- select and download the fonts you need
- install the fonts on your system

Linux

Installing new packages will require the *super admin* privileges.
Example, in Fedora:

```
sudo dnf install \
  google-noto-sans-cjk-jp-fonts \
  google-noto-serif-cjk-jp-fonts
```

In this example, we installed both *Serif* and *Sans-serif* typefaces of the CJK Japanese fonts.

In Fedora, the *Google Noto* fonts will be installed in folders matching this pattern: `/usr/share/fonts/google-noto*`

Other fonts

In this article, we use *Noto Fonts* because they are free, portable, and cover a variety of writing systems.

Other fonts of your choosing can be used in your documents. XeTeX should be able to use any fonts installed in your Operating System.

Part II
LaTeX

In the previous part, we talked about the fonts necessary to render CJK characters, and where to find them.

Next we'll need a working LaTeX environment…

Differences between TeX, LaTeX, and others

The original TeX was created in the late 1970s by Donald Knuth, who needed a new typesetting program.
At that time, Knuth was revising the second volume of his book "The Art of Computer Programming"*, got the galleys from his publisher, and was very disappointed in the result. The quality was so far below that of the first edition that he couldn't stand it. Around the same time, he saw a new book that had been produced digitally, and thought he could produce a digital typesetting system. So he started to learn about typography, type design, and the rules for typesetting math* (TUG, 2021)[7], and thus started his work on TeX.

> The idea behind TeX was *"to allow anybody to produce high-quality books with minimal effort, and to provide a system that would give exactly the same results on all computers, at any point in time"* (Wikipedia, 2022e)

The commands in TeX were basic, but allowed the creation of macros to extend the list of commands.

In the early 1980s, Leslie Lamport created LaTeX, a typesetting program written in the TeX macro language. (Wikipedia, 2022b) As such, LaTeX provides a large set of macros for TeX to interpret, and TeX is in charge of formatting the output. LaTeX packages are centralised in a repository called "The Comprehensive TeX Archive Network" (CTAN), *"the central place for all kinds of material around TeX"* (CTAN, 2022).

> Broadly speaking, you can think of LaTeX as: *"TeX, enhanced with a huge collection of macros: more than 6000 packages to date in CTAN (2022)."*

In 1989, Knuth declared that TeX was feature-complete, and only bug fixes would be made (Overleaf, 2022). Since then, new typesetting programs based on TeX appeared: pdfTeX, XeTeX, LuaTeX…

When those typesetting programs are used in conjunction with the LaTeX macros, we talk of: pdfLaTeX, XeLaTeX, LuaLaTeX…

The advantage of XeTeX (and therefore XeLaTeX) is that:

- XeTeX supports UTF-8 by default
- XeTeX can make use of the fonts that are installed on your computer (not just the standard LaTeX fonts)

This is required to handle texts in Japanese, Chinese, or other languages. We'll use XeLaTeX to generate our documents.

[7] I highly recommend you look at TUG (2021) if you want to learn more about the history of TeX

Getting LaTeX

The official LaTeX Project website (LaTeX, 2022) provides some information about LaTeX (including how to install LaTeX).

Link: `https://www.latex-project.org`

That being said, here is some information to get you started...

Windows®

On Windows®, I would recommend using MiKTeX, which includes (among other things):

- an *"integrated package manager"* (MiKTeX, 2022) (which will help you download the missing TeX packages, as you need them. This allows you to keep *"just enough TeX"* on your computer for your work)
- *TeXworks*, a *"simple TeX front-end program (working environment)"* (TeXworks, 2022)

When it comes to TeX editors, I have a preference for *TeXstudio*. The goal of *TeXstudio* is to *"make writing LaTeX as easy and comfortable as possible"* (TeXstudio, 2022).

On Windows, I would usually install the following:

- MiKTeX: `https://miktex.org`
- *TeXstudio*: `https://www.texstudio.org`

Linux

On Linux, I normally install `texlive` (as most Linux distribution provide it through their official repositories (TeXLive, 2022)), as well as an editor for TeX (usually *TeXstudio*).

texlive
: *Required*. TeX formatting system.

texstudio
: *Recommended*. A feature-rich editor for LaTeX documents.

To install these packages in Fedora (requires *super user* privileges):

```
$ sudo dnf install texlive texstudio
```

Additional TeX packages need to be installed separately.

Such packages normally have a name that starts with "`texlive-`"
(e.g. `texlive-mdframed`)

Installing fonts

This is a reminder (from the previous part) that we need to install some specialised fonts to be able to render CJK characters.
In the previous part, we also explained how to choose which fonts to install (based on our needs), and how to install them.

Please see *"Fonts"* for more details.

Additional packages

Some of our examples require additional packages.

My advice is to try and render the documents we provide as examples. If you come across any error, then install the missing packages. This will save you some disk space.

Windows

If you're using MiKTeX, then it will automatically handle package dependencies (MiKTeX will ask you permission before downloading the missing packages for you).

This allows to keep *"just enough TeX"* (i.e. only install the necessary packages). The downside is, the MiKTeX repositories can be significantly slower than those of major Linux distributions.

Linux

Under Linux, you will need to install the necessary LaTeX packages (installation requires *super user* privileges):

xeCJK
: *Required*
Support for CJK documents in XeLaTeX. (Wenchang Sun, 2019)
Package name (Fedora): `texlive-xecjk`

xpinyin
: *Optional*
Automatically add pinyin to Chinese characters. (Lee, 2020)
Package name (Fedora): `texlive-xpinyin`

`xpinyin` is a really impressive package! It contains a large database of Chinese characters and their pronunciation.

Documentation for `xeCJK` and `xpinyin`

To this day, the documentation for `xeCJK` (Wenchang Sun, 2019) and `xpinyin` (Lee, 2020) is available in Chinese only.

While "CJK" stands for *"Chinese, Japanese, Korean"*, only Chinese speakers have access to the documentation (which is very technical).
The `xpinyin` documentation is a bit easier to follow, despite the language barrier (it is reasonable to assume that `xpinyin` might be used by learners of Mandarin Chinese, who wish to quickly add *pinyin* to the texts they want to study).

Finding a document covering all the steps for rendering CJK characters in LaTeX (including fonts, package installation and use) was hard. Hopefully this article will help you getting set up.

I am focusing primarily on Japanese and Chinese scripts. That being said, some of the information contained in this document may apply to other writing systems.

Using CJK fonts in LaTeX

To manage the CJK fonts, we'll need to use the `xeCJK` package:

```
\usepackage{xeCJK}
```

We can set the CJK fonts to be used for the *whole* document. This must be done in the *preamble*:

```
\setCJKmainfont{Noto Serif JP}
\setCJKsansfont{Noto Sans JP}
\setCJKmonofont{Noto Sans Mono CJK JP}
```

This way, when `\sffamily` is used (e.g. inside a `\textsf{}` block), LaTeX will automatically render CJK characters with the *Noto Sans JP* font.

It is also possible to manually switch to a different CJK font:

```
\CJKfontspec{Noto Sans JP}
```

Finding the font names

In our document, we'll need to tell LaTeX which fonts we want to use. This implies referring to the font by name.

To find the font name, we have several solutions:

1. explore the directory where the *Noto fonts* are installed (e.g. `usr/share/fonts/google-noto-cjk`), open the font with a program like `gnome-font-viewer`, and copy the font name
2. open *LibreOffice* and view the list of available fonts, sorted by name

I know, it is rather ironic to use *LibreOffice* in order to find the font name we wish to use in a LaTeX document, but in many cases it might be more convenient this way.

Part III
Japanese texts in LaTeX

In the previous parts, we saw how to prepare our environment (including the type of fonts to use) to render CJK characters in LaTeX.

In the next two parts, we'll see how to add *ruby* characters for Japanese (*furigana*) and Chinese (*pinyin*).

Furigana in Japanese

In Japanese, the same group of *kanji* can have multiple pronunciations.

The main reason for this is *(warning: overly simplified version ahead...)* : Japan got its writing system from Chinese; the two languages are very different, there was not a 1-to-1 match between the two vocabularies, and the different *kanji* did not arrive all at once but in successive waves spread over several centuries...

Usually, a *kanji* in Japanese will have one *kun'yomi*[8] pronunciation, and one or more *on'yomi*[9] pronunciations. (Dexter, 2017)

While it is not uncommon for a *kanji* to have 3 or 4 possible pronunciations, a few characters may have a lot more than that...

To illustrate this phenomenon, here are some of the possible pronunciations for the character 日 (meaning "sun" or "day"). For clarity, I assigned a number to the different pronunciations...

[8] *kun'yomi*: the original, indigenous Japanese pronunciation
[9] *on'yomi*: a pronunciation derived from the Chinese

The multiple Japanese pronunciations of “日”

1	nichi 日	nichi	Day
2	ni hon 日本, nippon 日本	nihon, nippon	Japan
	yasumi no hi		
3	休みの日	yasumi no hi	Day off
	kinen bi		
4	記念日	kinenbi	Memorial day, commemoration day, anniversary
	nichi you bi		
(1,4)	日曜日	nichi youbi	Sunday
	mai nichi		
(1)	毎日	mai nichi	Everyday
	hi bi hi bi		
(3,4)	日々, 日日	hibi	Day after day, the every day
	dou jitsu		
5	同日	doujitsu	The same day
	rai jitsu		
(5)	来日	rai jitsu	(At a) later date
	ichi nichi ichi nichi		
(1)	一日, 1日	ichi nichi	One day (duration)
	ichi nichi (juu)		
	一日 (中)	ichi nichi (juu)	All day (long), throughout the day
	tsuitachi tsuitachi		
6	一日, 1日	tsuitachi	First day of the month
	tsukitachi		
7	一日 *(archaism)*	tsukitachi	The 1st day of the month
	futsuka futsuka		
8	二日, 2日	futsuka	The 2nd day of the month, 2 days
	…	…	…
	touka touka		
(8)	十日, 10日,	touka	The 10th day of the month, 10 days Also irregular: juu yokka 14日, ni juu yokka 24日
	ototoi		
9	一昨日	ototoi	The day before yesterday
	kinou		
10	昨日	kinou	Yesterday
	kyou		
11	今日	kyou	Today
	ashita		
12	明日	ashita	Tomorrow
	asu		
13	明日	asu	Tomorrow, (in the) near future
	asatte		
14	明後日	asatte	The day after tomorrow

Source: Takoboto (2022)

In this example, we counted no fewer than 14 sounds associated with the “日” character (note: there might be more…), and multiple ways to pronounce 日本, 一日, 明日.

> A particular set of *kanji* may have many possible pronunciations based on context. It is not always possible to know with certitude which pronunciation was the intended one (even though one is usually more likely than others). This is why the Japanese *furigana* need to be specified manually…

In the next part, we’ll see that Mandarin Chinese is very different in that regard…

Adding *furigana* in LaTeX

First you'll need to use the `ruby` package:

```
\usepackage{ruby}
```

Then you can modify some configuration. Here is what I used for this document:
`ruby-config.tex`

```
\renewcommand{\rubysize}{0.4}      % default:  0.4
\newcommand{\defaultrubysep}{0ex} % default: -0.5ex
\renewcommand{\rubysep}{\defaultrubysep}
```

The `ruby` package will give you access to the `\ruby` command, which you can summon like that:

```
\ruby{text}{ruby-characters}
```

Here is an example:

```
\ruby{明日}{あした}
```

And here is the result:

明日(あした)

Simplifying the process

Using the `\ruby` command is very easy to use, calling it for every character is very tedious (especially if you have a lot of text) so we'll define our own macro.

We will use the `\foreach` command from package `tikz` to make our life easier, and define a new command that we'll call `\furi` (to *furi*ously add *furigana*... maybe):

`furi.tex`

```
% This source code is under the BSD License
% Copyright 2022 Pierre S. Caboche

\usepackage{ruby,tikz}
\newcommand{\furi}[1]{\foreach \kanji/\furigana in {#1}{\ruby{\kanji}{\
    ↪ furigana\vphantom{あ}}}}
```

We'll see this command in action in our example but first, a word about `TikZ`...

TikZ is a user-friendly syntax layer for PGF (Portable Graphic Format), a macro package to create graphic elements in TeX. (Tantau, 2022)

The name TikZ is a recursive acronym which stands for *"TikZ ist kein Zeichenprogramm"*, which is German for *"TikZ is not a drawing tool"* (Wikipedia, 2022d) (the original developer of TikZ, Till Tantau, is from Germany, hence the name). We are using only a tiny subset of TikZ (namely, the `\foreach` command).

Here are the steps to quickly add *furigana* to some text:

First, copy the following template:

```
\furi{/,/}
```

Then add your text in Japanese:

```
\furi{栄光に向って走る\ あの列車に乗って行こう/,/}。。。
```

Cut the "`/,`" and paste it between each group of characters that may (or may not) require some *furigana*:

```
\furi{栄光/,に/,向/,って/,走/,る\ /,あの/,列車/,に/,
乗/,って/,行/,こう/}。。。
```

Add the *furigana*:

```
\furi{栄光/えいこう,に/,向/すか,って/,走/はし,る\ /,あの/,
列車/れっしゃ,に/,乗/の,って/,行/ゆ,こう/}。。。
```

And voilà! We're done!

Result

Here is what the LaTeX code looks like:

```
\furi{栄光/えいこう,に/,向/すか,って/,走/はし,る\ /,あの/,列車/れっしゃ,
    ↪ に/,乗/の,って/,行/ゆ,こう/}。。。
```

And here is what it prints:

栄光(えいこう)に向(すか)って走(はし)る あの列車(れっしゃ)に乗(の)って行(ゆ)こう。。。

Another example

Here is another example using our `\furi` macro:

example-02-furigana.tex

```
% This source code is under the BSD License
% Copyright 2022 Pierre S. Caboche

% Note: use XeLaTeX for rendering
%!TEX encoding = UTF-8

\documentclass{article}

\usepackage{xeCJK}

\usepackage{ruby,tikz}
\newcommand{\furi}[1]{\foreach \kanji/\furigana in {#1}{\ruby{\kanji}{\
    ↪ furigana\vphantom{あ}}}}

\renewcommand{\rubysize}{0.4}      % default:  0.4
\newcommand{\defaultrubysep}{0ex} % default: -0.5ex
\renewcommand{\rubysep}{\defaultrubysep}

\setCJKmainfont{Noto Sans JP}

\begin{document}
  \furi{明日/あした,がある さ/,明日/\textbf{あす},がある/}\par
  \furi{若/わか,い/,僕/ぼく,に/,は/わ,夢/ゆめ,がある/}\par
  \furi{いつかきっと、いつかきっと/}\par
  。。。
\end{document}
```

And here is how it renders:

明日(あした)があるさ明日(あす)がある
若(わか)い僕(ぼく)には夢(わ ゆめ)がある
いつかきっと、いつかきっと
。。。

Part IV
Chinese texts in LaTeX

In the previous parts, we saw the following:

- how to install LaTeX, for Windows and Linux
- how to find portable fonts for different writing systems
- how to use the CJK fonts in XeLaTeX
- how to add *ruby* characters to text

In our example, we saw how to add *furigana* to Japanese texts. This is the general technique, which can be used for adding *ruby* characters to texts in any language or dialect.

However, if your goal is to add the *pinyin* pronunciation to text in Mandarin Chinese (Simplified or Traditional), then there is a LaTeX package to do that: `xpinyin`.

Automatically add *pinyin*

As we saw in the previous part, a *kanji* in Japanese can have multiple pronunciations. In Chinese, however, things are different. With a few exceptions, most *kanji* in Mandarin Chinese have only one possible pronunciation.

Thanks to this (almost) 1-to-1 match, it is possible to know (with a high degree of confident) how a given *kanji* should be pronounced in Mandarin Chinese. The `xpinyin` package was developed to automatically add the *pinyin*.

As mentioned earlier, there are some exceptions…
For example, the character 乐 may be pronounced “lè” as in 快乐 (kuài lè) (happy), or “yuè” as in 音乐 (yīn yuè) (music). When that happens, it is possible to specify the pronunciation manually with the `\xpinyin` macro (examples below).

The `xpinyin` package

To use the `xpinyin` package, add the following in your document preamble:

```
\usepackage{xpinyin}
```

The `xpinyin` package allows to add *pinyin* in your LaTeX document.

Here is a quick introduction to some of its features:

`\pinyin`
: It is used to output pinyin. For the convenience of input, ü can be replaced by v.

  Examples:
  `\pinyin{lv2zi}` → lǘ zi
  `\pinyin{nv3hai2zi}` → nǚ hái zi

Below is an example of each of the “four tones” (+neutral tone) of Mandarin Chinese:

`\pinyin{ma1}` → mā `\pinyin{ma2}` → má `\pinyin{ma3}` → mǎ
`\pinyin{ma4}` → mà `\pinyin{ma}` → ma

`\xpinyin`

`\xpinyin` can be used to set pinyin.
This is useful within `\xpinyin*` macro or `pinyinscope` environment.
Example:

`\xpinyin{乐}{yue4}` → yuè 乐

`\xpinyin*`

Automatic phonetic notation for Chinese characters Example:

`\xpinyin*{甄士隐梦幻识通灵}` → zhēn shì yǐn mènghuàn shí tōng líng 甄士隐梦幻识通灵

`\xpinyin*{\xpinyin{重}{zhong4}要}` → zhòng yào 重要

`pinyinscope`

Automatic phonetic notation for Chinese characters in the `pinyinscope` environment (useful for long texts).
Syntax:

```
\begin{pinyinscope}[ options ]
  ...
\end{pinyinscope}
```

For the rest of the documentation (in Chinese only), see: Lee (2020) at `https://ctan.org/pkg/xpinyin`

Example

Below is a complete example:

example-03-pinyin.tex

```
% This source code is under the BSD License
% Copyright 2022 Pierre S. Caboche

% Note: use XeLaTeX for rendering
%!TEX encoding = UTF-8

\documentclass{article}

\usepackage{xeCJK}
\usepackage{xpinyin}
\setCJKmainfont{Noto Sans CJK SC} % Simplified Chinese

\begin{document}
  \begin{itemize}
    \item \xpinyin*{妈麻马骂吗} $\leftarrow$ the ``four tones" (+neutral
        ↪ tone) of Mandarin Chinese
    \item \xpinyin*{快乐的音乐} $\leftarrow$ the \emph{pinyin} on the
        ↪ last character is wrong
    \item \xpinyin*{快乐的音\xpinyin{乐}{yue4}} $\leftarrow$ this is
        ↪ correct (we specified the last \emph{pinyin} manually)
  \end{itemize}

  \bigskip

  \begin{pinyinscope}
You can also use the \texttt{pinyinscope} environment:\par
\bigskip
滚滚长江东逝水，浪花淘尽英雄。\par
是非成败转头空，青山依旧在，几度夕阳红。\par
白发渔樵江渚上，惯看秋月春风。\par
一壶浊酒喜相逢，古今多少事，都付笑谈中。\par
\bigskip
是非成败转头空，青山依旧在，惯看秋月春风。\par
一壶浊酒喜相逢，古今多少事，滚滚长江东逝水，浪花淘尽英雄。\par
几度夕阳红。白发渔樵江渚上，都付笑谈中。。。。。。\par
  \end{pinyinscope}
\end{document}
```

And here is the result:

Text with *pinyin*

- mā má mǎ mà ma
  妈麻马骂吗 ← the “four tones” (+neutral tone) of Mandarin Chinese
- kuài lè de yīn lè
  快乐的音乐 ← the *pinyin* on the last character is wrong
- kuài lè de yīn yuè
  快乐的音乐 ← this is correct (we specified the last *pinyin* manually)

You can also use the `pinyinscope` environment:

gǔn gǔn zhǎngjiāngdōng shì shuǐ làng huā táo jǐn yīng xióng
滚滚长江东逝水，浪花淘尽英雄。
shì fēi chéng bài zhuǎn tóu kōng qīng shān yī jiù zài jǐ dù xī yánghóng
是非成败转头空，青山依旧在，几度夕阳红。
bái fā yú qiáo jiāng zhǔ shàng guàn kàn qiū yuè chūn fēng
白发渔樵江渚上，惯看秋月春风。
yī hú zhuó jiǔ xǐ xiāng féng gǔ jīn duō shǎo shì dōu fù xiào tán zhōng
一壶浊酒喜相逢，古今多少事，都付笑谈中。

shì fēi chéng bài zhuǎn tóu kōng qīng shān yī jiù zài guàn kàn qiū yuè chūn fēng
是非成败转头空，青山依旧在，惯看秋月春风。
yī hú zhuó jiǔ xǐ xiāng féng gǔ jīn duō shǎo shì gǔn gǔn zhǎngjiāngdōng shì shuǐ làng huā táo jǐn yīng xióng
一壶浊酒喜相逢，古今多少事，滚滚长江东逝水，浪花淘尽英雄。
jǐ dù xī yánghóng bái fā yú qiáo jiāng zhǔ shàng dōu fù xiào tán zhōng
几度夕阳红。白发渔樵江渚上，都付笑谈中。。。。。。

Part V
LaTeX vs. *LibreOffice*, *Word*®…

I started to use LaTeX because of how easy it is to add *ruby* characters. After experimenting with *LibreOffice* and *Word*®, I tried LaTeX and was genuinely impressed by how much time it saved me for this task…

To properly grasp how effective LaTeX is at adding *ruby* characters, we need to understand how the same process is done in *LibreOffice* and *Word*®.

LibreOffice

Configuration

This is a quick guide to enabling *ruby* character support in *LibreOffice*.

Language settings

- Go to *Tools* > *Options…*
- under *Language settings* > *Languages*
  - check the *Asian* box
  - select a default language (e.g. *Japanese*)
- Click *OK*

This should enable the support for features like *Asian Phonetic Guide*

Edit the context menu

- Go to *Tools* > *Customize…*
  This allow you to edit the Context Menu (accessible when you right-click on something)
- Under *Target* (to the right)
  - Select *Text*
    This allows you to edit the Context Menu for when you right-click on some selected text
- Under *Search* (to the top-left)
  - Search for *“Asian Phonetic Guide”*
    Asian Phonetic Guide should appear in the list of *Available Commands*
  - Move *Asian Phonetic Guide* from *Available Commands* (the menu on the left) to *Assigned Commands* (the menu on the right)
- Click *OK*

From now on, *Asian Phonetic Guide* should be available when you right click on some selected text.

Adding *pinyin* with *LibreOffice*

- Select some text
- Right-click
  If you've followed the configuration above, then *"Asian Phonetic Guide…"* should appear in the menu
- Select *"Asian Phonetic Guide…"*
  A menu called *"Asian Phonetic Guide"* should appear

The *"Asian Phonetic Guide"* menu allows you to specify the *ruby* characters you want to add to the text. You also need to specify the alignment, position, and style for the *ruby* characters.

A very slow process…

In *LibreOffice*, you will need to repeat the above process for virtually ***every*** character for which you want to add some *ruby* annotation.

This is slow and tedious…

Not only that, but should you wish to change the size or alignment of the *ruby* characters, then you will also have to repeat the process for ***every*** character in your document.

Word®

The process is somewhat similar in *Microsoft*® *Word*®.

Microsoft® *Word*® suffers from the same quirk as *LibreOffice*, where you need to specify the *ruby* (and style, size, alignment) for nearly ***every*** character in your document…

Word® has one advantage over *LibreOffice*: for each selected character, *Word*® suggests a possible *ruby*… but LaTeX does even better (at least in Mandarin) by automatically adding *pinyin* to your text, thanks to the `xpinyin` package (and LaTeX does it for free!)

The fact that LaTeX also produces beautiful documents is just the icing on the cake…

Part VI
Conclusion

I started to use LaTeX out of necessity, because I needed a way to write documents (for my personal use) which would contain a lot of Japanese *furigana* (*ruby*). After some experimentation, I found that adding *ruby* characters was considerably more efficient in LaTeX than in either *LibreOffice* or *Word*® (see *Part V*).

The process of using LaTeX also turned out to be easier than I initially thought. LaTeX produces very good-looking documents by default, which you can customise if you need to (like I did in this article and others…)
But the main reason for me to use LaTeX was the possibility to define custom commands which you can then *re-use* at will, to produce documents with a consistent layout (this, in my opinion, is one of LaTeX's biggest strengths).

So this is, in a nutshell, how I started out with LaTeX…

And everything was fine, until I tried to render my LaTeX files on a different Operating System…

Indeed, the way I wrote my LaTeX files prevented them from properly render on Linux (which I'm now using more and more, including at home). Solving this issue was the subject of *Parts I and II* (which apply not only to Japanese and Chinese, but any language that use a non-Latin writing script).

At first I was focused on documents that feature texts in Japanese language, and how to add *furigana* (Part III). Then I discovered the `xpinyin` package, which does much of the same but is tailored towards Standard Mandarin Chinese. I found it to be a related issue, and therefore decided to cover it as well (in Part IV).

Hopefully this document will have helped you discover what LaTeX can do, and how to handle languages like Japanese, Chinese, or Korean (and other non-Latin writing scripts).

What we learned

This article presented the following subjects:

- fonts and non-latin writing systems
- Noto fonts
- *ruby* characters, *furigana*, *pinyin*
- some aspects specific to the Japanese and Chinese languages
- the `ruby` package
- the `xpinyin` package
- the use of custom macros (to add *furigana*)

References

CTAN (2022). The Comprehensive TeX Archive Network. https://www.ctan.org.

Dexter, K. (2017). On'yomi and kun'yomi in kanji: what's the difference? https://www.tofugu.com/japanese/onyomi-kunyomi/.

Google (2022). Noto Fonts: Beautiful and free fonts for all languages. https://www.google.com/get/noto/.

LaTeX (2022). The LaTeX project. https://www.latex-project.org.

Lee, Q. (2020). xpinyin – Automatically add pinyin to Chinese characters. https://ctan.org/pkg/xpinyin.

MiKTeX (2022). The MiKTEX project. https://miktex.org.

Overleaf (2022). The TEX family tree: LATEX, pdfTEX, XƎTEX, LuaTEX and ConTEXt. https://www.overleaf.com/learn/latex/Articles/The_TeX_family_tree:_LaTeX,_pdfTeX,_XeTeX,_LuaTeX_and_ConTeXt.

Takoboto (2022). Takoboto: Japanese dictionary & Nihongo learning tool. https://takoboto.jp/?q=%E6%97%A5.

Tantau, T. (2022). pgf – A Portable Graphic Format for TeX. https://github.com/pgf-tikz/pgf.

TeXLive (2022). TEXLive. *TeX Users Group*. https://tug.org/texlive/.

TeXstudio (2022). TEXstudio. https://www.texstudio.org.

TeXworks (2022). TeXworks. *TeX Users Group*. http://www.tug.org/texworks/.

TUG (2021). Just what is TEX? *TeX Users Group*. http://www.tug.org/whatis.html.

Wenchang Sun, Leo Liu, Q. L. (2019). xecjk – Support for CJK documents in XeLaTeX. https://ctan.org/pkg/xecjk.

Wikipedia (2022a). Half-width kana. https://en.wikipedia.org/wiki/Half-width_kana.

Wikipedia (2022b). LaTeX. https://en.wikipedia.org/wiki/LaTeX.

Wikipedia (2022c). Noto Fonts. https://en.wikipedia.org/wiki/Noto_fonts.

Wikipedia (2022d). PGF/TikZ. https://en.wikipedia.org/wiki/PGF/TikZ.

Wikipedia (2022e). TeX. https://en.wikipedia.org/wiki/TeX.

Index

\furi *(custom macro)*, 22

ConTeXt, 31

furigana, 2, 3, 20, 30

Google Noto fonts, 2, 8, 9, 14

half-width kana, 12
hiragana, 3

kana, 12
kanji, 3, 20, 21, 25
katakana, 3, 12
kun’yomi, 20

LibreOffice, 1, 2, 19, 28–30
Linux, 1, 2, 9, 16, 30
LuaLaTeX, 15
LuaTeX, 15, 31

Meiryo, 2, 8
Microsoft®, 1, 8, 29
MiKTeX, 16, 17, 31
monospaced, 11, 12

on’yomi, 20

pdfLaTeX, 15
pdfTeX, 15, 31
pinyin, 2, 3, 20, 25, 29, 30
portability, 8

ruby, 2, 3, 28–30

sans-serif, 9, 11
serif, 9, 11

TeXstudio, 16
TeXworks, 16
TikZ, 22, 23
tofu, 8

Windows®, 1, 2, 16
Word®, 1, 2, 28–30

xeCJK, 18
XeLaTeX, 15, 17, 25
XeTeX, 14, 15, 31
xpinyin, 2, 17, 18, 29