The LuaTEX-ja package

The LuaTEX-ja project team

September 7, 2011

Contents

This documentation is far from complete. It may have many grammatical (and contextual) errors.

Part I
User's manual

0,0,0.

1 Introduction

The LuaTEX-ja package is a macro package for typesetting high-quality Japanese documents in LuaTEX.

1.1 Backgrounds

Traditionally, ASCII pTEX, an extension of TEX, and its derivatives are used to typeset Japanese documents in TEX. pTEXis an engine extension of TEX: so it can produce high-quality Japanese documents without using very complicated macros. But this point is a mixed blessing: pTEX is left behind from other extensions of TEX, especially ε-TEX and pdfTEX, and from changes about Japanese processing in computers (*e.g.*, the UTF-8 encoding).

Recently the extensions of pTEX, namely upTEX (Unicode-implementation of pTEX) and ε-pTEX (Merging of pTEXand ε-TEX extension), have developed to fill those gap to some extent, but gaps are still exist.

However, the appearance of LuaTEX changed the whole situation. With using Lua 'callbacks', users can customize the internal processing of LuaTEX. So there is no need to modify sources of the TEX engine to support Japanese typesetting: to do this, we only have to write Lua script for appropriate callbacks.

1.2 Major Changes from pTEX

The LuaTEX-ja package is much influenced by pTEX engine. The initial target of development was to implement features of pTEX. However, *LuaTEX-ja is not a just porting of pTEX: Unnatural specifications/behaviors of pTEX were not adopted.*

The followings are major changes from pTEX:

- Japanese fonts are a tuple of a 'real' font, a Japanese font metric (**JFM**, for short), and an optional string called 'variation'.
- In pTEX, a linebreak after Japanese character is ignored (and doesn't yield a space), since Japanese texts can linebreak almost everywhere. However, LuaTEX-ja doesn't have this function completely, because of a specification of LuaTEX.
- The insertion process of glues/kerns between two Japanese characters and between a Japanese character and other characters (we refer these glues/kerns as **JAglue**) is rewritten from scratch.
  - As LuaTEX's internal character handling is 'node-based' (*e.g.*, `of{}fice` doesn't prevent ligatures), the insertion process of **JAglue** is now 'node-based'.
  - Furthermore, nodes between two characters which have no effects in linebreak (*e.g.*, `\special` node) are ignored in the insertion process.
  - In the process, two Japanese fonts which differ in their 'real' fonts only are identified.
- At the present, vertical typesetting (*tategaki*), is not supported in LuaTEX-ja.

For detailed information, see Part III.

1.3 Notations

In this document, the following terms and notations are used:

- Characters are divided into two types:
  - **JAchar**: standing for Japanese characters such as Hiragana, Katakana, Kanji and other punctuation marks for Japanese.'
  - **ALchar**: standing for all other characters like alphabets.
- A word in sans-serif font (like prebreakpenalty) represents an internal parameter for Japanese typesetting, and it is used as a key in `\ltjsetparameter` command.
- The word "primitive" is used not only for primitives in LuaTEX, but also for control sequences that defined in the core module of LuaTEX-ja.

1.4 About the project

Project Wiki `http://sourceforge.jp/projects/luatex-ja/wiki/FrontPage%28en%29`
This project is hosted by SourceForge.JP.

Members

2 Getting Started

2.1 Installation

To install the LuaTEX-ja package, you will need:

- LuaTEX, version 0.65.0-beta or later.
  If you are using TEX Live 2011 or W32TEX, you don't have to worry.
- The source archive of LuaTEX-ja, of course :)

The installation methods are as follows:

1. Download the source archive.

   At the present, LuaTEX-ja has no official release, so you have to retrieve the archive from the repository. You can retrieve the Git repository via

   ```
   $ git clone git://git.sourceforge.jp/gitroot/luatex-ja/luatexja.git
   ```

   or download the archive of HEAD in the master branch from

   `http://git.sourceforge.jp/view?p=luatex-ja/luatexja.git;a=snapshot;h=HEAD;sf=tgz`.

2. Extract the archive. You will see `src/` and several other sub-directories.
3. Copy all the contents of `src/` into one of your `TEXMF` tree.
4. If `mktexlsr` is needed to update the filename database, make it so.

2.2 Cautions

- The encoding of your source file must be UTF-8.
- conflicts with unicode-math

2.3 Using in plain TEX

To use LuaTEX-ja in plain TEX, simply put the following at the beginning of the document:

```
\input luatexja.sty
```

This does the minimal setting (like `ptex.tex`) for typesetting Japanese documents:

- The following 6 Japanese fonts are preloaded.

classification	**font name**	**13.5 Q**	**9.5 Q**	**7 Q**
mincho	Ryumin-Light	`\tenmin`	`\sevenmin`	`\fivemin`
gothic	GothicBBB-Medium	`\tengt`	`\sevengt`	`\fivegt`

  - The 'Q' is an unit used in Japanese phototypesetting, and $1\,\mathrm{Q} = 0.25\,\mathrm{mm}$. This length is stored in a dimension `\jQ`.
  - It is widely accepted that the font 'Ryumin-Light' and 'GothicBBB-Medium' aren't embedded into PDF files, and the PDF reader substitutes them by some external Japanese font. We adopt this custom to the default setting.
  - size

- A character in Unicode is treated as **JAchar** if and only if its code-point has more than or equal to U+0100.
- The amount of glue that are inserted between **JAchar** and **ALchar** (the parameter `xkanjiskip`) is set to

$$0.25\,\texttt{\textbackslash zw}^{+1\,\mathrm{pt}}_{-1\,\mathrm{pt}} = \frac{27}{32}\,\mathrm{mm}^{+1\,\mathrm{pt}}_{-1\,\mathrm{pt}}.$$

  Here `\zw` is a counterpart of `em` for Japanese fonts, that is, the length of 'full-width' in the current Japanese font.

2.4 Using in LaTeX

LaTeX 2ε Using in LaTeX 2ε is basically same. To set up the minimal environment for Japanese, you only have to load `luatexja.sty`:

```
\usepackage{luatexja}
```

It also does the minimal setting (the counterpart in pLaTeX is `plfonts.dtx` and `pldefs.ltx`):

- `JY3` is used as the font encoding for Japanese fonts (in horizontal direction).
  If vertical typesetting is supported by LuaTeX-ja, `JT3` will be used for vertical fonts.
- Two font families `mc` and `gt` are defined:

classification	family	`\mdseries`	`\bfseries`	scale
mincho	`mc`	Ryumin-Light	GothicBBB-Medium	0.960444
gothic	`gt`	GothicBBB-Medium	GothicBBB-Medium	0.960444

- Japanese characters in math mode are typeset by the font family `mc`.

However, the above setting is not sufficient for Japanese-based documents. To do this, You are better to use class files other than `article.cls`, `book.cls`, ... The better alternatives are:

- BXjscls
- ltjarticle, ltjbook?
- ltjsarticle, ltjsbook?

2.5 Changing Fonts

Remark: Japanese Characters in Math Mode Since pTeX supports Japanese characters in math mode, there are sources like the following:

```
$f_{高温}$~($f_{\text{high temperature}}$).
\[ y=(x-1)^2+2\quad{}よって\quad y>0 \]
$5\in{}素:=\{\,p\in\mathbb N:\text{$p$ is a
    prime}\,\}$.
```

$f_{高温}$ ($f_{\text{high temperature}}$).

$$y=(x-1)^2+2\quad よって\quad y>0$$

$5\in 素:=\{\,p\in\mathbb N:\text{p is a prime}\,\}$.

We (the project members of LuaTeX-ja) think that using Japanese characters in math mode are allowed if these are used as identifiers. In this point of view,

- The lines 1 and 2 above are not correct, since '高温' in above is used as a textual label, and 'よって' is used as a conjunction.
- However, the line 3 is correct, since '素' is used as an identifier.

Hence, in our opinion, the above input should be corrected as:

```
$f_{\text{高温}}$~%
($f_{\text{high temperature}}$).
\[ y=(x-1)^2+2\quad
  \mathrel{\text{よって}}\quad y>0 \]
$5\in{}素:=\{\,p\in\mathbb N:\text{$p$ is a
    prime}\,\}$.
```

$f_{\text{高温}}$ ($f_{\text{high temperature}}$).

$$y=(x-1)^2+2\quad\mathrel{\text{よって}}\quad y>0$$

$5\in 素:=\{\,p\in\mathbb N:\text{p is a prime}\,\}$.

どう繋ごうか？

In this chapter, we don't describe how to change Japanese fonts in math mode. For the method, please see Part II.

plain TeX To change Japanese fonts in plain TeX, you must use the primitive `\jfont`. So please see Part II.

NFSS2 For LaTeX 2ε, LuaTeX-ja simply adopted font selection system from that of pLaTeX 2ε (in: `plfont.dtx`).

- Two control sequences `\mcdefault` and `\gtdefault` are used to specify the default font family for *mincho* and *gothic*, respectively.
- Commands `\fontfamily`, `\fontseries`, `\fontshape` and `\selectfont` can be used to change attributes of Japanese fonts.

	encoding	**family**	**series**	**shape**
alphabetic fonts	`\romanencoding`	`\romanfamily`	`\romanseries`	`\romanshape`
Japanese fonts	`\kanjiencoding`	`\kanjifamily`	`\kanjiseries`	`\kanjishape`
both	—	–	`\fontseries`	`\fontshape`
auto select	`\fontencoding`	`\fontfamily`	—	—

- For defining a Japanese font family, use `\DeclareKanjiFamily` instead of `\DeclareFontFamily`.

fontspec To use with `fontspec` package, it is needed to load `luatexja-fontspec` package in the preamble. This additional package automatically loads `luatexja` and `fontspec` package, if needed.

In `luatexja-fontspec` package, the following 4 commands are defined as counterparts of original commands in `fontspec`:

Japanese fonts	`\jfontspec`	`\setmainjfont`	`\setsansjfont`	`\newjfontfamily`
alphabetic fonts	`\fontspec`	`\setmainfont`	`\setsansfont`	`\newfontfamily`

Note that there is no command named `\setmonojfont`, since it is popular for Japanese fonts that (nearly) all Japanese glyphs have the same width.

3 Changing Parameters

There are many parameters in LuaTeX-ja. And due to the implementation, most of them were not stored as internal register of TeX, but as an original storage system in LuaTeX-ja. Hence, to change or recall those parameters, you have to use commands `\ltjsetparameter` and `\ltjgetparameter`.

3.1 Editing the range of JAchar

As noted before, the default setting is:

A character in Unicode is treated as **JAchar** if and only if its code-point has more than or equal to U+0100.

↑ TODO: CHANGE THIS!

3.2 kanjiskip and xkanjiskip

JAglue is divided into the following three categories:

- Glues/kerns specified in JFM. If `\inhibitglue` is issued, this glue will be not inserted.
- The default glue which inserted between two **JAchar**s (kanjiskip).
- The default glue which inserted between a **JAchar** and an **ALchar** (xkanjiskip).

The value (a skip) of kanjiskip or xkanjiskip can be changed as the following.

```
\ltjsetparameter{kanjiskip={0pt plus 0.4pt minus 0.4pt},
                 xkanjiskip={0.25\zw plus 1pt minus 1pt}}
```

It may occur that JFM contains the data of 'ideal width of kanjiskip' and/or 'ideal width of xkanjiskip'. To use these data from JFM, set the value of kanjiskip or xkanjiskip to `\maxdimen`.

3.3 Insertion Setting of xkanjiskip

It is not desirable that xkanjiskip is inserted between every boundary between **JAchar** and **ALchar**. For example, xkanjiskip should not be inserted after opening parenthesis (*e.g.*, compare '(あ' and '(あ').

LuaTEX-ja can control whether xkanjiskip can be inserted before/after a character, by using jaxspmode and alxspmode parameters.

```
\ltjsetparameter{jaxspmode={`あ,preonly},
    alxspmode={`\!,postonly}}
p あ q !う
```

p あq !う

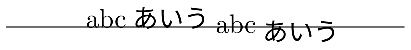
The second argument `preonly` means 'the insertion of xkanjiskip is allowed before this character, but not after'. the other possible values are `postonly`, `allow` and `inhibit`.

If you want to enable/disable all insertion of kanjiskip and xkanjiskip, set autospacing and autoxspacing parameters to `false`, respectively.

3.4 Shifting Baseline

To make a match between a Japanese font and an alphabetic font, sometimes the shifting of baseline of one of the pair. In pTEX, this is achieved by setting `\ybaselineshift` to a non-zero length (the baseline of alphabetic fonts is shifted below). However, for documents whose main language is not Japanese,it is good to shift the baseline of Japanese fonts, but not that of alphabetic fonts. Because of this, LuaTEX-ja can be independently set the shifting amount of the baseline of alphabetic fonts (yalbaselineshift parameter) and that of Japanese fonts (yjabaselineshift parameter).

```
\vrule width 150pt height 0.4pt depth 0pt\hskip
    -120pt
\ltjsetparameter{yjabaselineshift=0pt,
    yalbaselineshift=0pt}abc あいう
\ltjsetparameter{yjabaselineshift=5pt,
    yalbaselineshift=2pt}abc あいう
```



Here the horizontal line in above is the baseline of a line.

There is an interesting side-effect from that the baseline of Japanese fonts can be shifted: characters in different size can be vertically aligned center in a line, by setting two parameters appropriately.

```
xyz 漢字
{\scriptsize
  \ltjsetparameter{yjabaselineshift=-1pt,
    yalbaselineshift=-1pt}
  XYZ ひらがな
}abc かな
```

xyz 漢字 XYZ ひらがな abc かな

3.5 'tombow'

'tombow' is a mark for indicating 4 corners and horizontal/vertical center of the paper. pLATEXand this LuaTEX-ja support 'tombow' by their kernel. The following steps are needed to typeset tombow:

1. First, define the banner which will be printed at the upper left of the paper. This is done by assigning a token list to `\@bannertoken`.

   For example, the following sets banner as '`filename (2012-01-01 17:01)`':

   ```
   \makeatletter

   \hour\time \divide\hour by 60 \@tempcnta\hour \multiply\@tempcnta 60\relax
   \minute\time \advance\minute-\@tempcnta
   \@bannertoken{%
      \jobname\space(\number\year-\two@digits\month-\two@digits\day
      \space\two@digits\hour:\two@digits\minute)}%
   ```

2. ...

Part II

Reference

4 Font Metric and Japanese Font

4.1 `\jfont` primitive

To load a font as a Japanese font, you must use the `\jfont` primitive instead of `\font`, while `\jfont` admits the same syntax used in `\font`. LuaTEX-ja automatically loads `luaotfload` package, so TrueType/OpenType fonts with features can be used for Japanese fonts:

```
\jfont\tradgt={file:ipaexg.ttf:script=latn;%
  +trad;jfm=ujis} at 14pt
\tradgt{}当／体／医／区
```

當／體／醫／區

Note that the defined control sequence (`\tradgt` in the example above) using `\jfont` is not a *font_def* token.

Prefix Besides `file:` and `name:` prefixes, `psft:` can be used a prefix in `\jfont` (and `\font`) primitive.. Using this prefix, you can specify a font that has its name only and is not related to any real font.

The typical use of this `psft:` prefix is ...

Features

4.2 Structure of JFM file

A JFM file is a Lua script which has only one function call:

`luatexja.jfont.define_jfm { ... }`

Real data are stored in the table which indicated above by `{ ... }`. So, the rest of subsection are devoted to describe the structure of this table. Note that all lengths in a JFM file are floating-point numbers in design-size unit.

`dir=`⟨*direction*⟩ (required)

The direction of JFM. At the present, only `'yoko'` is supported.

`zw=`⟨*length*⟩ (required)

The amount of the length of the 'full-width.

`zh=`⟨*length*⟩ (required)

`kanjiskip={`⟨*natural*⟩`,` ⟨*stretch*⟩`,` ⟨*shrink*⟩`}` (optional)

This field specifies the 'ideal' amount of kanjiskip. As noted in Subsection 3.2, if kanjiskip is `\maxdimen`, the value specified in this field is used (if this field is not specified in JFM, 0 pt is used). Note that ⟨*stretch*⟩ and ⟨*shrink*⟩ fields are in design-size unit too.

`xkanjiskip={`⟨*natural*⟩`,` ⟨*stretch*⟩`,` ⟨*shrink*⟩`}` (optional)

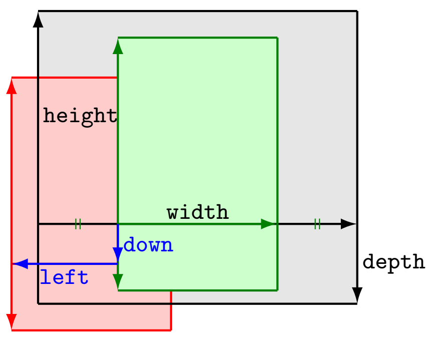
Like the `kanjiskip` field, this field specifies the 'ideal' amount of xkanjiskip.

Besides from above fields, a JFM file have several sub-tables those indices are natural numbers. The table indexed by $i \in \omega$ stores informations of 'character class' i. At least, the character class 0 is always present, so each JFM file must have a sub-table whose index is `[0]`. Each sub-table (its numerical index is denoted by i) has the following fields:

`chars={`⟨*character*⟩`, ...}` (required except character class 0)

This field is a list of **JAchar**s which are in this character type i. This field is not required if $i = 0$, since all **JAchar** which are not in any character class other than 0 (hence, the character class 0 contains most of **JAchar**s). In the list, a **JAchar** can be specified by its code number, or by the character itself (as a string of length 1).

In addition to those 'real' characters, the following 'imaginary characters' can be specified in the list:



Consider a node containing Japanese character whose value of the `align` field is `'middle'`.

- The black rectangle is a frame of the node. Its width, height and depth are specified by JFM.
- Since the `align` field is `'middle'`, the 'real' glyph is centered horizontally (the green rectangle).
- Furthermore, the glyph is shifted according to values of fields `left` and `down`. The ultimate position of the real glyph is indicated by the red rectangle.

Figure 1: The position of the 'real' glyph

`width=`⟨*length*⟩`,` `height=`⟨*length*⟩`,` `depth=`⟨*length*⟩`,` `italic=`⟨*length*⟩

Specify width of characters in character class i, height, depth and the amount of italic correction. These fields are required.

`left=`⟨*length*⟩`,` `down=`⟨*length*⟩`,` `align=`⟨*align*⟩

These are for adjusting the position of the 'real' glyph. Legal values of `align` field are `'left'`, `'middle'` and `'right'`. If one of these 3 fields are omitted, `left` and `down` are treated as 0, and `align` field is treated as `'left'`. The effects of these 3 fields are indicated in Figure 1.

In most cases, `left` and `down` fields are 0, while it is not uncommon that the `align` field is `'middle'` or `'right'`. For example, setting the `align` field to `'right'` is practically needed when the current character class is the class for opening delimiters'.

`kern={}`

`glue={}`

4.3 Math Font Family

5 Parameters

5.1 `\ltjsetparameter` primitive

5.2 List of Parameters

In the following list of parameters,

- '$*$' : local
- '†' always global
- No mark: the last of paragraph

kcatcode ={⟨*chr_code*⟩,⟨*value*⟩}

prebreakpenalty ={⟨*chr_code*⟩,⟨*penalty*⟩}

postbreakpenalty ={⟨*chr_code*⟩,⟨*penalty*⟩}

jatextfont ={⟨*jfam*⟩,⟨*jfont_cs*⟩}

jascriptfont ={⟨*jfam*⟩,⟨*jfont_cs*⟩}

jascriptscriptfont ={⟨*jfam*⟩,⟨*jfont_cs*⟩}

yjabaselineshift =⟨*dimen*⟩*

yalbaselineshift =⟨*dimen*⟩*

jaxspmode ={⟨*chr_code*⟩,⟨*mode*⟩}

alxspmode ={⟨*chr_code*⟩,⟨*mode*⟩}

autospacing =⟨*bool*⟩*

autoxspacing =⟨*bool*⟩*

kanjiskip =⟨*skip*⟩

xkanjiskip =⟨*skip*⟩

jcharwidowpenalty =⟨*penalty*⟩

differentjfm =⟨*mode*⟩†

jacharrange =⟨*ranges*⟩*

6 Other Primitives

7 Control Sequences for LaTeX 2ε

Part III
Implementations

8 Storing Parameters

8.1 Used Dimensions and Attributes

Here the following is the list of dimension and attributes which are used in LuaTeX-ja.

`\jQ` (dimension)

`\jH` (dimension)

`\ltj@zw` (dimension)

`\ltj@zh` (dimension)

`\jfam` (attribute)

`\ltj@curjfnt` (attribute) The font index of current Japanese font.

`\ltj@charclass` (attribute) The character class of Japanese *glyph_node*.

`\ltj@yablshift` (attribute) The amount of shifting the baseline of alphabetic fonts in scaled point (2^{-16} pt).

`\ltj@ykblshift` (attribute) The amount of shifting the baseline of Japanese fonts in scaled point (2^{-16} pt).

`\ltj@autospc` (attribute) Whether the auto insertion of kanjiskip is allowed at the node.

`\ltj@autoxspc` (attribute) Whether the auto insertion of xkanjiskip is allowed at the node.

`\ltj@icflag` (attribute) For distinguishing 'kinds' of the node. To this attribute, one of the following value is assigned:

ITALIC (1)
PACKED (2)
KINSOKU (3)
FROM_JFM (4)
LINE_END (5)
KANJI_SKIP (6)
XKANJI_SKIP (7)
PROCESSED (8)
IC_PROCESSED (9)
BOXBDD (15)

`\ltj@kcat`*i* (attribute) Where *i* is a natural number which is less than 8. These 8 attributes store bit vectors indicating ...

8.2 Stack System of LuaTEX-ja

Overview LuaTEX-ja has its own stack system, and most parameters of LuaTEX-ja are stored in it. To clarify the reason, imagine the parameter kanjiskip is stored by a skip, and consider the following source:

```
\ltjsetparameter{kanjiskip=0pt}ふがふが.%
\setbox0=\hbox{\ltjsetparameter{kanjiskip=5pt}ほげほげ}
\box0.ぴよぴよ\par
```

ふがふが. ほ げ ほ げ. ぴよぴよ

As described in Part II, the only effective value of kanjiskip in an hbox is the latest value, so the value of kanjiskip which applied in the entire hbox should be 5 pt. However, by the implementation method of LuaTEX, this '5 pt' cannot be known from any callbacks. In the `tex/packaging.w` (which is a file in the source of LuaTEX), there are the following codes:

```
void package(int c)
{
    scaled h;                    /* height of box */
    halfword p;                  /* first node in a box */
    scaled d;                    /* max depth */
    int grp;
    grp = cur_group;
    d = box_max_depth;
    unsave();
    save_ptr -= 4;
    if (cur_list.mode_field == -hmode) {
        cur_box = filtered_hpack(cur_list.head_field,
                                 cur_list.tail_field, saved_value(1),
                                 saved_level(1), grp, saved_level(2));
        subtype(cur_box) = HLIST_SUBTYPE_HBOX;
```

Notice that `unsave` is executed *before* `filtered_hpack` (this is where `hpack_filter` callback is executed): so '5 pt' in the above source is orphaned at +unsave+, and hence it can't be accessed from `hpack_filter` callback.