LuaTEX-ja 宏包

LuaTEX-ja 项目团队

目 次

第 I 编
用户手册

1　引言

LuaTEX-ja 宏包是应用于 LuaTEX 引擎上的高质量日语文档排版宏包。

1.1 开发背景

一般情况下，TEX 下的日语文档输出，是 ASCII pTEX（TEX 的一个扩展）及其衍生软件来完成的。pTEX 作为 TEX 的一个扩展引擎，在生成高质量的日语文档时，规避了繁杂的宏编写。但是在和同时期的引擎相比之下，pTEX 的处境未免有些尴尬：pTEX 已经远远落后于 ε-TEX 和 pdfTEX，此外也没有跟上计算机上对日文处理的演进（比如，UTF-8 编码，TrueType 字体，OpenType 字体）。

最近开发的 pTEX 扩展，即 upTEX（Unicode 下的 pTEX 实现）和 ε-pTEX（pTEX 和 ε-TEX 的融合版本），虽然在部分情况上弥补了上述的差距，但是差距依然存在。

不过，LuaTEX 的出现改变了整个状况。用户可以通过使用 Lua 语言的“callback”来调整 LuaTEX 的内部处理机制。所以，没有必要去通过修改引擎的源代码来支持日文排版，相反，我们需要做的仅仅是编写其当处理 callback 的 Lua 脚本。

1.2 与 pTEX 的差异所在

LuaTEX-ja 宏包在设计上，受 pTEX 影响很大。最初开发的主要议题是实现 pTEX 的特性。不过，**LuaTEX-ja 不是简简单单的移植 pTEX，很多不自然的特征和现象都被移出了。**

下面列举出了一些和 pTEX 的差异：

- 一个日文字体是由三部分构成的元组：实际的字体（如小塚明朝，IPA 明朝），日文字体测度（**JFM**）和变体字串。
- pTEX 中，日文字符之后的断行并不允许（也不产生空格），其他在源码中的断行是可以随处允许的。不过，因为 LuaTEX 的特殊关系，LuaTEX-ja 并没有这个功能。
- 插在日文字符和其他字符言之间的胶/出格（我们将此称为 **JAglue**）是重现实现的。
  - 在 LuaTEX 中，内部的字符处理是“基于 node 的”（例如：`of{}fice` 不会避免合字），**JAglue** 的插入处理，现在也是“基于 node 的”。
  - 此外，两个字符之间的 node 在断行时不起作用的（例如，`\special`node），还有意大利体校正带来的出格在插入处理中也是被忽略的。
  - **警告：鉴于以上两点，在 pTEX 中分割 JAglue 处理的多种方法不再生效。**明确地说，下列两种方法不再生效：

    ちょ`{}`っと　　ちょ`\/`っと

    如果想得到此种结果，请使用空盒子替代：

    ちょ`\hbox{}`っと
  - 处理过程中，两个在“真实”字体上具区别的日文字体可以被识别出来。
- 当下，LuaTEX-ja 并不支持直行排版。

1.3 一些约定

在本文档中，有下面一些约定：

- 所有的日文字符为 **JAchar**，所有的其他字符为 **ALchar**
- primitive，该词在本文档中不仅表示 LuaTEX 的基本控制命令，也包括 LuaTEX-ja 的相关的基本控制命令
- 所有的自然数从 0 开始

- 本文档中文文档是根据日文文档翻译并添加部分材料组织而成的，所以在大部分篇幅上还是主要依赖于日文原文档

1.4 关于本项目

项目 wiki 本项目 wiki 正在不断编写中。

- http://sourceforge.jp/projects/luatex-ja/wiki/FrontPage%28en%29（英文）
- http://sourceforge.jp/projects/luatex-ja/wiki/FrontPage（日文）
- http://sourceforge.jp/projects/luatex-ja/wiki/FrontPage%28zh%29（中文）

开发者

- 北川弘典
- 前田一贵
- 八登崇之
- 黒木裕介
- 阿部纪行
- 山本宗宏
- 本田知亮
- 斋藤修三郎
- 马起园

2 初次使用

2.1 安装

在安装 LuaTEX-ja 之前，请确认：

- LuaTEX（版本号为大于 0.65）和相关支持宏包。
  如果用户使用的是 TEXLive2011 以及最新版本的 W32TEX，都可以不考虑此项。
- LuaTEX-ja 的源码:)
- xunicode 宏包，当前版本必须为 *v0.981(2011/09/09)*。
  如果你使用 fontspec 宏包，xunicode 必须存在。但是请注意该包版本，其他版本可能不会正常工作。

安装方法如下：

1. 用下列诸多方法下载源码归档。当前，LuaTEX-ja 并无**稳定**版本。
   - 从 git 中复制
     `$ git clone git://git.sourceforge.jp/gitroot/luatex-ja/luatexja.git`
   - 下载 `master` 分支 HEAD 的 `tar.gz` 归档
     http://git.sourceforge.jp/view?p=luatex-ja/luatexja.git;a=snapshot;h=HEAD;sf=tgz
   - 现在 LuaTEX-ja 已经包含于 CTAN 中（在 `macros/luatex/generic/luatexja` 文件夹）和 W32TEX 中（`luatexja.tar.gz`）。这些版本基于 `master` 版本。
2. 解压归档。你将看到 `src/`和其他文件夹。但是只有 `src/`文件夹的内容是 LuaTEX-ja 宏包运行所需。
3. 将 `src/`文件夹内容复制到 `TEXMF` 树下某个位置。`TEXMF/tex/luatex/luatexja/`是一个例子。如果你克隆了全部的 Git 仓库，将 `src/`做成符号链接来代替复制也不错。
4. 如果需要升级文件名数据库，请执行 `mktexlsr`。

2.2 警告

- 你的源文件编码必须是 UTF-8。其他编码不行，例如 EUC-JP 或者 Shift-JIS。

2.3 plain TEX 格式下的使用

在 plain TEX 下使用 LuaTEX-ja 相当简易，在文档开头放置一行：

```
\input luatexja.sty
```

这里做出了做小的日文文档排版设定（如 `ptex.tex`）：

- 提前加载了六种日文字体，如下：

字体	字体名	‘10 pt’	‘7 pt’	‘5 pt’
明朝体	Ryumin-Light	`\tenmin`	`\sevenmin`	`\fivemin`
哥特体	GothicBBB-Medium	`\tengt`	`\sevengt`	`\fivegt`

  - “Q（级）”是日本照排中使用的尺寸单位，1Q = 0.25mm。该长度保存在长度`\jQ` 中。
  - 广为接受的“Ryumin-Light”和“GothicBBB-Medium”字体不嵌入 PDF 文件，而 PDF 阅读器则会使用外部日文字体替代（例如，在 Adobe Reader 中使用 Kozuka Mincho 字体替代 Ryumin-Light）。我们使用默认设定。
  - 一般情况下，相同大小日文字体比西文字体要大一下。所以实际的日文字体尺寸需哟小于西文字体，即使用一个缩放率：0.962216。

- 在 **JAchar** 和 **ALchar** 之间插入的胶（xkanjiskip 参数）大小为：

$$(0.25 \times 0.962216 \times 10\mathrm{pt})^{+1\mathrm{pt}}_{-1\mathrm{pt}} = 2.40554\mathrm{pt}^{+1\mathrm{pt}}_{-1\mathrm{pt}}$$

2.4 在 LATEX 下使用

LATEX 2ε 在LATEX 2ε下使用基本相同。设定日文的最小环境，你只需加载 `luatexja.sty`：

```
\usepackage{luatexja}
```

这些做了最小的设定（作用相当于 pLATEX 中的`plfonts.dtx` 和`pldefs.ltx`）：

- `JY3` 是日文字体编码（在水平方向）。
  在将来 LuaTEX-ja 要支持直行排版的时候，`JT3` 会用于直行字体。

- 定义了两个字体族：`mc` 和`gt`：

字体	字体族	\mdseries	\bfseries	缩放率
mincho	mc	Ryumin-Light	GothicBBB-Medium	0.962216
gothic	gt	GothicBBB-Medium	GothicBBB-Medium	0.962216

  注意的是两个字体族的粗体系列同为中等系列的**哥特**族。这 pLATEX 中的规定。在近些年中的 DTP 实务中有仅使用 2 个字体的趋向（是为 Ryumin-Light 和 GothicBBB-Medium）。

- 在数学模式下，日文字符使用`mc` 字体族来排印

不过，上述设定并不能满足排版基于日文的文档。为了排印基于日文的文档，你最好不要使用 `article.cls`，`book.cls` 等文档类文件。现在，我们有相当于 `jclasses`（pLATEX 标准文档类）和 `jsclasses`（奥村晴彦）的文档类，即 `ltjclasses` 和 `ltjsclasses`。

OTF 包中的\CID，\UTF 及其他宏　pLaTeX 下，otf 宏包（斎藤修三郎开发）是用来排印存在于 Adobe-Japan1-6 但不存在于 JIS X 0208 中的字符。该包已经广泛使用，LuaTeX-ja 支持部分 otf 包中的部分功能。如果你想使用这些功能，加载 luatexja-otf 宏包。

```
森\UTF{9DD7}外と内田百\UTF{9592}とが\UTF{9AD9}島屋に行く。

\CID{7652}飾区の\CID{13706}野家,
葛飾区の吉野家
```

森鷗外と内田百閒とが髙島屋に行く。
葛飾区の吉野家，葛飾区の吉野家

2.5　改变字体

注记：数学模式下的日文字符　pTeX 支持在数学模式下的日文字符，如以下源码：

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

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

$$y=(x-1)^2+2 \quad よって \quad y>0$$

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

我们（LuaTeX-ja 项目成员）认为在数学模式下使用日文字符，只有在这些字符充当标识符时才是正确的。在这点下，

- 第 1 行和第 2 行是不正确的，因为“高温”的作用为文本标签，“よって”用作为连词。
- 不过，第 3 行是正确的，因为“素”是作为标识符的。

那么，根据我们的观点，上述输入应当校正为：

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

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

$$y=(x-1)^2+2 \quad よって \quad y>0$$

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

我们也认为使用日文字符作为标识符的情况极为少见，所以我们不在此章节描述如何在数学模式下改变日文字体。关于此方法，请参见。

plain TeX　在 plain TeX 下改变日文字体，你必须使用基本语句\jfont。请参见。

NFSS2　对于LaTeX 2$_\varepsilon$，LuaTeX-ja 采用了 pLaTeX 2$_\varepsilon$中（即 plfonts.dtx）大部分字体选择系统。

- \mcdefault 和\gtdefault 控制语句用来分别控制默认的 *mincho* 和 *gothic* 字体族。默认值: mc 用于\mcdefault，gt 用于\gtdefault。
- 命令\fontfamily，\fontseries，\fontshape 个\selectfont 用来改变日文字体属性。

	编码	族	系列	形状	选择
西文字体	\romanencoding	\romanfamily	\romanseries	\romanshape	\useroman
日文字体	\kanjiencoding	\kanjifamily	\kanjiseries	\kanjishape	\usekanji
两者	—	–	\fontseries	\fontshape	—
自动选择	\fontencoding	\fontfamily	—	—	\usefont

\fontencoding{<encoding>}依赖于参数以改变西文字体或者日文字体。例如，\fontencoding{JY3}改变当前日文字体至 JY3，\fontencoding{T1}改变西文字体至 T1。\fontfamily 也会改变日文字体或西文字体的族，抑或二者。细节详见。

- 对于定义日文字体族，使用\DeclareKanjiFamily 代替\DeclareFontFamily。不过，在现在的实现中，使用\DeclareFontFamily 不会引起任何问题。

2.6 fontspec

为与 fontspec 宏包共存，需要在导言区中使用 luatexja-fontspec 宏包。这个附加宏包会自动加载 luatexja 和 fontspec。

在 luatexja-fontspec，定义了如下七条命令，这些命令和 fontspec 的相关命令对比如下：

日文字体	\jfontspec	\setmainjfont	\setsansjfont	\newjfontfamily
西文字体	\fontspec	\setmainfont	\setsansfont	\newfontfamily
日文字体	\newjfontface	\defaultjfontfeatures	\addjfontfeatures	
西文字体	\newfontface	\defaultfontfeatures	\addfontfeatures	

```
\fontspec[Numbers=OldStyle]{TeX Gyre Termes}
\jfontspec{IPAexMincho}
JIS~X~0213:2004→辻

\addjfontfeatures{CJKShape=JIS1990}
JIS~X~0208:1990→辻
```

JIS X 0213:2004 →辻
JIS X 0208:1990 →辻

请注意并没有\setmonofont 命令，因为流行的日文字体几乎全部是等宽的。另注意，出格特性在这 7 个命令中默认关闭，因为此特性会与 **JAglue** 冲突（参见）。

3 参数设定

LuaTEX-ja 包含大量的参数，以控制排版细节。设定这些参数需要使用命令: \ltjsetparameter 和\ltjgetparameter 命令。

3.1 JAchar 范围的设定

在设定 **JAchar** 之前，需要分配一个小于 217 的自然数。如：

```
\ltjdefcharrange{100}{"10000-"1FFFF,`漢}
```

请注意这个设定是全局性的，不建议在文档正文中进行设定。

在范围设定好了之后，需要进行jacharrange 的设定：

```
\ltjsetparameter{jacharrange={-1, +2, +3, -4, -5, +6, +7, +8}}
```

这里定义了 8 个范围，在每个范围之前使用“+”或“-”进行设定，其中如果为 −，则代表该范围为 **ALchar**，如果为 +，则该范围视作 **JAchar**。

LuaTEX-ja 默认设定了 8 个范围，这些范围来源于下列数据：

- Unicode 6.0
- Adobe-Japan1-6 与 Unicode 之间的映射Adobe-Japan1-UCS2
- 八登崇之的 upTEX 宏包：PXbase

范围 8[J] ISO 8859-1（Latin-1 补充）的上半部和 JIS X 0208（日文基本字符集）的重叠部分，包含下列字符：

- § (U+00A7, Section Sign)
- ¨ (U+00A8, Diaeresis)
- ° (U+00B0, Degree sign)
- ± (U+00B1, Plus-minus sign)
- ´ (U+00B4, Spacing acute)
- ¶ (U+00B6, Paragraph sign)
- × (U+00D7, Multiplication sign)
- ÷ (U+00F7, Division Sign)

范围 1[A] 包含于 Adobe-Japan1-6 中的拉丁字符，此范围包含下列 Unicode 区域，但不包括上述提到过的范围 8:

- U+0080–U+00FF: 拉丁字母补充-1
- U+0100–U+017F: 拉丁字母扩充-A
- U+0180–U+024F: 拉丁字母扩充-B
- U+0250–U+02AF: 国际音标扩充
- U+02B0–U+02FF: 进格修饰符元
- U+0300–U+036F: 组合音标附加符号
- U+1E00–U+1EFF: 拉丁字母扩充附加

范围 2[J] 希腊文和西里尔字母，使用 JIS X 0208 的大部分日文字体包含这些字符:

- U+0370–U+03FF: 希腊字母
- U+0400–U+04FF: 西里尔字母
- U+1F00–U+1FFF: 希腊文扩充

范围 3[J] 标点以及杂项符号:

- U+2000–U+206F: 一般标点符号
- U+2070–U+209F: 上标及下标
- U+20A0–U+20CF: 货币符号
- U+20D0–U+20FF: 符号用组合附加符号
- U+2100–U+214F: 类字母符号
- U+2150–U+218F: 数字形式
- U+2190–U+21FF: 箭头符号
- U+2200–U+22FF: 数学运算符号
- U+2300–U+23FF: 杂项技术符号
- U+2400–U+243F: 控制图像
- U+2500–U+257F: 制表符
- U+2580–U+259F: 区块元素
- U+25A0–U+25FF: 几何形状
- U+2600–U+26FF: 杂项符号
- U+2700–U+27BF: 什锦符号
- U+2900–U+297F: 补充性箭头-B
- U+2980–U+29FF: 混合数学符号-B
- U+2B00–U+2BFF: 杂项符号和箭头符号
- U+E000–U+F8FF: 私用区域

范围 4[A] 通常情况下不包含于日文字体的部分。本范围包含有其他范围尚未涵盖部分。故，我们直接给出定义:

```
\ltjdefcharrange{4}{%
   "500-"10FF, "1200-"1DFF, "2440-"245F, "27C0-"28FF, "2A00-"2AFF,
  "2C00-"2E7F, "4DC0-"4DFF, "A4D0-"A82F, "A840-"ABFF, "FB50-"FE0F,
  "FE20-"FE2F, "FE70-"FEFF, "FB00-"FB4F, "10000-"1FFFF} % non-Japanese
```

范围 5[A] 代替以及补充私有使用区域。

范围 6[J] 日文字符。

- U+2460–U+24FF: 圈状字母数字
- U+2E80–U+2EFF: CJK 部首补充
- U+3000–U+303F: CJK 标点符号
- U+3040–U+309F: 平假名

- U+30A0–U+30FF: 片假名
- U+3190–U+319F: 汉文标注号
- U+31F0–U+31FF: 片假名音标补充
- U+3200–U+32FF: 圈状 CJK 字母及月份
- U+3300–U+33FF: CJK 兼容
- U+3400–U+4DBF: CJK 统一表意文字扩充 A
- U+4E00–U+9FFF: CJK 统一表意文字
- U+F900–U+FAFF: CJK 兼容表意文字
- U+FE10–U+FE1F: 直行标点
- U+FE30–U+FE4F: CJK 兼容形式
- U+FE50–U+FE6F: 小写变体
- U+20000–U+2FFFF: （补充字符）

范围 7[J] 不包含于 Adobe-Japan1-6 的 CJK 字符。

- U+1100–U+11FF: 谚文字母
- U+2F00–U+2FDF: 康熙部首
- U+2FF0–U+2FFF: 汉字结构描述字符
- U+3100–U+312F: 注音字母
- U+3130–U+318F: 谚文兼容字母
- U+31A0–U+31BF: 注音字母扩充
- U+31C0–U+31EF: CJK 笔划
- U+A000–U+A48F: 彝文音节
- U+A490–U+A4CF: 彝文字母
- U+A830–U+A83F: 一般印度数字
- U+AC00–U+D7AF: 谚文音节
- U+D7B0–U+D7FF: 谚文字母扩充-B

3.2 kanjiskip 和 xkanjiskip

JAglue 分为下列三类范畴：

- JFM 设定的胶或出格值。如果在一个日文字符附近使用`\inhibitglue`，则胶便不会插入。
- 两个 **JAchar** 之间默认插入的胶（kanjiskip）
- **JAchar** 和 **ALchar** 之间默认插入的胶（xkanjiskip）

kanjiskip 和 xkanjiskip 的设定如下所示：

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

当 JFM 包含“kanjiskip 理想宽度”和/或“xkanjiskip 理想宽度”数据时，上述设定产生作用。如果想用 JFM 中的数据，请设定 kanjiskip 或 xkanjiskip 为`\maxdimen`。

3.3 xkanjiskip 插入设定

并不是在所有的 **JAchar** 和 **ALchar** 周围插入 xkanjiskip 都是合适的。比如，在开标点之后插入 xkanjiskip 并不合适 [如，比较“(あ”和“(あ”]。LuaTEX-ja 可以通过设定 **JAchar** 的 jaxspmode 以及 **ALchar** 的 alxspmode 来控制 xkanjiskip 在字符前后的插入。

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

第二个参数 preonly 表示的含义为“允许在该字符前插入 xkanjiskip，但不允许在该字符之后插入”。其他参数还有 postonly，allow 和 inhibit。[TODO]

用户如果想开启/关闭 kanjiskip 和 xkanjiskip 的插入，设定 autospacing 和 autoxspacing 参数为 true/false 即可。

3.4 基线浮动

为了确保日文字体和西文字体能够对其,有时需要浮动其中一者的基线。在pTEX中,此项设定由设定\yabaselineshift为非零长度（西文字体基线应向下浮动)。不过，如果文档的中主要语言不是日文，那么最好上浮日文字体的基线，西文字体不变。如上所述，LuaTEX-ja 可以独立设定西文字体的基线（yabaselineshift 参数）和日文字体的基线（yjabaselineshift 参数)。

```
\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
```

abc あいう abc あいう

上述水平线为此行基线。

这里还有一个有趣的副作用：不同大小的字符可以通过适当调整这两个参数而在一行中垂直居中。下面是一个例子（注意，参数值并没有精心调整)：

```
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 裁剪框标记

裁剪框标记是在一页的四角和水平/垂直中央放置的标记。在日文中，裁剪框被称为“トンボ”。pLATEX 和 LuaTEX-ja 均在底层支持裁剪框标记。需要下列步骤来实现：

1. 首先，首先定义页面左上角将会出现的注记。这由向@bannertoken 分配一个 token 列完成。
   例如，下列所示将会设定注记为“filename (YYYY-MM-DD hh:mm)”：

```
\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. [TODO]

第 II 编
参考指南

4 字体测度和日文字体

4.1 \jfont 基本语句

为了加载日文字体，需要使用\jfont 基本语句替代\font，前者支持后者所有相同句法。LuaTEX-ja 自动加载 luaotfload 宏包，故 TrueType/OpenType 字体的特性可以使用于日文字体：

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

當／體／醫／區

注意定义的控制序列 (上例中的\tradgt) 使用的\jfont 并不是一个 *font_def* 标记, 故类似\fontname\tradgt 输入会引起错误。我们指出定义于\jfont 控制序列>

JFM

注：kern 特性

4.2 \psft 前缀

除使用 file: 和 name: 外, 我们还可以在\jfont（以及\font）中使用 psft: 来设定一个“名义上”的并不嵌入 PDF 中的日文字体。此前缀的典型使用是定义“标准”日文字体, 即“Ryumin-Light”和“GothicBBB-Medium”。

cid 键 默认使用 psft: 前缀定义的字体是为 Adobe-Japan1-6 CID 字体。也可以使用 cid 键来使用其他的 CID 字体，如中文和韩文。

```
\jfont\testJ={psft:Ryumin-Light:cid=Adobe-Japan1-6;jfm=jis}     % 日语
\jfont\testD={psft:Ryumin-Light:jfm=jis}                        % 默认值为Adobe-Japan1-6
\jfont\testC={psft:AdobeMingStd-Light:cid=Adobe-CNS1-5;jfm=jis} % 繁体中文
\jfont\testG={psft:SimSun:cid=Adobe-GB1-5;jfm=jis}              % 简体中文
\jfont\testK={psft:Batang:cid=Adobe-Korea1-2;jfm=jis}           % 韩文
```

注意上述代码使用了 jfm-jis.lua，这可以用于日文字体，也可以用于中文和韩文字体。

当下，LuaTEX-ja 只支持如上例中的 4 个值，改为其他值，例如:

```
\jfont\test={psft:Ryumin-Light:cid=Adobe-Japan2;jfm=jis}
```

会发生下列错误:

```
! Package luatexja Error: bad cid key `Adobe-Japan2'.

See the luatexja package documentation for explanation.
Type  H <return>  for immediate help.
<to be read again>
                   \par
l.78

? h
I couldn't find any non-embedded font information for the CID
`Adobe-Japan2'. For now, I'll use `Adobe-Japan1-6'.
Please contact the LuaTeX-ja project team.
?
```