# LuaCJK 宏包
# The LuaCJK package*

Mingyu Xia†

Released 2026-11-22 v0.0.4

**摘要 | Abstract**

LuaCJK 为 LuaTEX 提供中文排版。它把当前的 CJK 字体记录在一个 LuaTEX 属性里，并在文字定形之前把每个 CJK 字形放进该字体。

LuaCJK provides Chinese typesetting for LuaTEX. It records the current CJK font in a LuaTEX attribute and sets each CJK glyph in that font before the text is shaped.

## 目录 Contents

**1 简介** **4**

**2 加载** **4**

**3 字体** **5**

3.1 子区块 . . . . . 7
3.2 后备字体 . . . . . 7
3.3 基线调整 . . . . . 8
3.4 加载 Fandol . . . . . 8
3.5 伪粗体与伪斜体 . . . . . 9
3.6 数学模式中的汉字 . . . . . 10

*https://ctan.org/pkg/luacjk, https://github.com/myhsia/LuaCJK
†xiamingyu@westlake.edu.cn

**4 文本排版** **11**
4.1 间距 . . . 11
4.2 标点 . . . 12
4.3 断行 . . . 15
4.4 暂停与钩子 . . . 16
4.5 抄录与 listings . . . 17
4.6 装饰 . . . 18

**5 与 xeCJK 的比较** **21**
5.1 行为差异 . . . 21
5.2 未提供的功能 . . . 22

**1 Introduction** **23**

**2 Loading** **23**

**3 Fonts** **25**
3.1 Sub-blocks . . . 26
3.2 Fallback fonts . . . 27
3.3 Baseline shift . . . 28
3.4 Loading Fandol . . . 28
3.5 Fake bold and slant . . . 28
3.6 CJK in mathematics . . . 29

**4 Text layout** **30**
4.1 Spacing . . . 30
4.2 Punctuation . . . 32
4.3 Line breaking . . . 36
4.4 Suspending and hooks . . . 37
4.5 Verbatim and listings . . . 38
4.6 Decorations . . . 39

**5 Comparison with xeCJK** **43**
5.1 Different behavior . . . 43
5.2 Not available . . . 43

**A Implementation** **45**
A.1 Loading and conflicts . . . . . 45
A.2 Registers, styles, and options . . . . . 46
A.3 Fonts . . . . . 70
A.4 Names and configuration . . . . . 90
A.5 The Lua module . . . . . 93
A.6 Settings and data . . . . . 96
A.7 Node rewriting . . . . . 110
A.8 The passes . . . . . 123
A.9 Decorations . . . . . 165
A.10 Listings and ctex . . . . . 206

**Index** **214**

**Change History** **233**

# 1 简介

LuaCJK 是一个只用于 LuaTeX 的中文排版宏包。它的主要接口是 \setCJKmainfont、\setCJKsansfont、\setCJKmonofont、\setCJKfamilyfont、\CJKfamily 与 \LuaCJKsetup。

当前的 CJK 字体记录在一个 LuaTeX 属性里，每个 CJK 字形在文字定形之前就放进该字体。

# 2 加载

\usepackage[⟨*options*⟩]{LuaCJK}

LuaCJK 需要 LuaTeX 引擎。它与 luatexja 冲突：无论先后加载哪一个，都会报错。

\LuaCJKversion 打印 LuaCJK 的版本。

\LuaCJKsetup \LuaCJKsetup {⟨*key-value list*⟩}

配置 LuaCJK。这些键就是随后各节中记录的那些键。

这些键分为两组。

描述文字如何排版的设置只作用于当前分组，并且记录在字符本身上，所以分组中的文字即使在该分组已经结束之后才断行，也仍然保留它的设置：

```
\vbox{\begingroup\LuaCJKsetup{CJKglue = 10pt}中中\endgroup\par}
```

用的就是 10pt 的胶，尽管分组在 \par 之前已经关闭。这一组包含

- CJKglue, CJKecglue, CJKspace 与 xCJKecglue;
- PunctStyle, PunctKinsoku, AllowBreakBetweenPuncts, PunctWidth 与 PunctBoundWidth;
- LatinPunct 与 CJKmath;
- CheckSingle, WidowPenalty 与 CJLineBreak;
- AutoFallBack 与 Verb;
- 两个命令 \LuaCJKCancelSubCJKBlock 与 \LuaCJKRestoreSubCJKBlock。

其余的键是全局的，或者在加载宏包时起作用：

- KaiMingPunct, MiddlePunct, LongPunct 与 NoBreakLongPunct 修改字符集合；
- AutoFakeBold, AutoFakeSlant, EmboldenFactor 与 SlantFactor 在注册字族时起作用；
- LocalConfig 在加载宏包时起作用， LoadFandol 在 \begin{document} 处起作用。

声明 \LuaCJKDeclareCharClass、\LuaCJKDeclarePunctClass、 \LuaCJKDeclareSubCJKBlock、与 \LuaCJKsetkern 同样是全局的，这一点在各自的说明里都提到了。

同样的键也可以作为宏包选项给出：

```
\usepackage[PunctStyle = banjiao]{LuaCJK}
```

LocalConfig = ⟨**true**|*false*|*name*⟩ 从配置文件读取本地设置。 true 加载 luacjk.cfg， ⟨*name*⟩ 加载 luacjk-⟨*name*⟩.cfg，而 false（初始值）什么都不加载。文件在宏包末尾读取，所以它可以使用 LuaCJK 定义的每一个命令。文件不存在时只给出警告。

要在 ctex 中使用 LuaCJK，把它的 engine 选项设为 ⟨*luacjk*⟩：

```
\documentclass[engine = luacjk]{ctexart}
```

ctex 于是载入 LuaCJK 的 ctex-engine-luacjk.def 而不是它自己的 LuaTEX 后端，这个文件同时载入宏包，LuaCJK 不必再单独载入。

LuaCJK 与 LATEX 的标记 PDF（tagged PDF）项目协同工作。文档开头声明

```
\DocumentMetadata{tagging = on}
```

时，装饰的墨迹标记为 artifact，装饰符号不会出现在复制和选择的文本中；selectable = true 保留符号的文字。在 l3doc 文档中还要载入 latex-lab-testphase-l3doc，本手册就是这样生成的。选区的高度由 PDF 阅读器按字体的 hhea 上伸部和下伸部决定，对 Noto Serif CJK SC 是 1.437em，所以选区比 1em 的方块更高；这不是 LuaCJK 能改变的，也不影响字形本身。

## 3 字体

主要的字体命令接受一个可选的 fontspec 选项列表。选项既可以写在字体名之前，也可以写在字体名之后。无论西文使用哪一种编码，CJK 字体都在 Unicode 编码下选取，所以用 fontenc 把西文设为 T1 之类的旧编码不会影响中文。

\setCJKmainfont
\setCJKsansfont
\setCJKmonofont

\setCJKmainfont [⟨*options*⟩] {⟨*font*⟩} [⟨*options*⟩]

为罗马、无衬线与等宽字族设置 CJK 字体。

```
\setCJKmainfont[Scale = 1.2]
  {FandolSong-Regular.otf}
English 中文字体\\
\setCJKsansfont[Scale = 1.2]
  {FandolKai-Regular.otf}
\sffamily English 中文字体
```

English 中文字体
English 中文字体

\setCJKfamilyfont `\setCJKfamilyfont{⟨family⟩}[⟨options⟩]{⟨font⟩}[⟨options⟩]`

定义一个具名的 CJK 字族。字族可以用 `\CJKrmdefault` 之类的名字给出。

\CJKfamily `\CJKfamily {⟨family⟩}`

切换到具名的 CJK 字族。字族可以用 `\CJKrmdefault` 之类的名字给出。

```
\CJKfamily{zhsong}English 中文字体\\
\CJKfamily{zhhei}English 中文字体\\
\CJKfamily{zhkai}English 中文字体\\
\CJKfamily{zhfs}English 中文字体
```

English 中文字体
English 中文字体
English 中文字体
English 中文字体

\newCJKfontfamily `\newCJKfontfamily[⟨family⟩]⟨command⟩[⟨options⟩]{⟨font⟩}[⟨options⟩]`

定义一个切换到新声明字族的命令。与 `\newfontfamily` 一样，该命令只作用于当前分组。选项可以写在字体名之前，也可以写在之后。

\CJKfontspec `\CJKfontspec [⟨options⟩] {⟨font⟩}`

为当前分组选择一个 CJK 字体。与 `\fontspec` 一样，选项写在字体名之前：这个命令用在正文中间，若它还要读取字体名之后的方括号，正文紧接着的那个方括号就会被当成选项而消失。

`\songti`
`\heiti`
`\fangsong`
`\kaishu`
`\lishu`
`\youyuan`

名为 zhsong、zhhei、zhfs、zhkai、 zhli 与 zhyou 的字族的快捷方式。

```
\newCJKfontfamily[kai]\mykai
  {FandolKai-Regular.otf}
\mykai English 中文字体\\
\songti English 中文字体
```

English 中文字体
English 中文字体

\CJKrmdefault
\CJKsfdefault
\CJKttdefault
\CJKfamilydefault

由 \rmfamily、\sffamily 与 \ttfamily 选择的 CJK 字族，以及由 \normalfont 选择的字族。\CJKfamilydefault 默认跟随 \familydefault：主字族是三者之一时，它就是对应的那一族，否则它取 \CJKrmdefault。它在文档开始时选取，所以在导言区重定义它或 \familydefault，正文第一行就用新的字族。

### 3.1 子区块

一种 CJK 字体很少覆盖 Unicode 的每一个区块。子区块让一个范围之内的字符在一个字族里拥有自己的字体；这个区块随后成为字体命令的一个选项。

\LuaCJKDeclareSubCJKBlock \LuaCJKDeclareSubCJKBlock {⟨*block*⟩} {⟨*range list*⟩}

用给定的范围声明 ⟨*block*⟩，写法与 \LuaCJKDeclareCharClass 相同。这个名字随后成为一个字体选项，正如下例所示：设置了它的字族对该区块的字符使用第二个字体，没有设置它的字族则把这些字符留在主字体里。重叠的区块按声明顺序解决。

```
\LuaCJKDeclareSubCJKBlock{Kana}{3040 -> 30FF}
\setCJKmainfont[Kana = FandolKai-Regular.otf]
  {FandolSong-Regular.otf}
汉字 かんじ
```

汉字かんじ

\LuaCJKCancelSubCJKBlock \LuaCJKCancelSubCJKBlock {⟨*block list*⟩}
\LuaCJKRestoreSubCJKBlock \LuaCJKRestoreSubCJKBlock {⟨*block list*⟩}

在当前分组的剩余部分关闭或重新打开列出的区块。因为字体的选择是按字体标识符记录的，所以提供同一字体的两个字族共享同一个设置：最后选择该字体的字族说了算。

### 3.2 后备字体

一种 CJK 字体很少覆盖需要的每一个字符。当所选的 CJK 字体没有某个字符的字形时，后备字体按声明的顺序依次尝试。

\setCJKfallbackfamilyfont `\setCJKfallbackfamilyfont {⟨family⟩} [⟨options⟩] {⟨font⟩} [⟨options⟩]`

把 ⟨*font*⟩ 加入 ⟨*family*⟩ 的后备列表。字族可以用 `\CJKrmdefault` 之类的名字给出。多次调用该命令可以建立一条链；第一个拥有该字符的字体胜出。这条链按文字的字重、字形与字号解析，所以粗体里的字符取自后备字体的粗体字面，放大后的字符取自对应字号的字面。打开 `CJKmath` 时，排在数学模式里的汉字也走同一条链：列表取自负则这个数学字体的 CJK 字族，字号取数学样式的字号。

`AutoFallBack=`⟨*true*|**false**⟩ 是否使用声明过的后备字体。这个开关像间距的那些键一样跟随文字，所以可以只在一个分组里打开它。

```
\setCJKmainfont{FandolSong-Regular.otf}
\setCJKfallbackfamilyfont{\CJKrmdefault}
  {FandolKai-Regular.otf}
\LuaCJKsetup{AutoFallBack = true}中あ
```

中あ

### 3.3 基线调整

两种 CJK 字体很少是为彼此设计的：混排在同一行里，一种字体的字总显得偏高或偏低。一种字体可以带一个 `Baseline` 值，把它的字整体向上或向下移动，使两种字体立在同一条基线上。这个值属于字体而不属于文字，所以它是声明这个字体的那条命令的一个选项。

`Baseline=`⟨*dimension*⟩ 把这个字体的字向上移动 ⟨*dimension*⟩，负值则向下移动。它和其他字体选项一样可以写在后备字体的声明里，这也是它最常见的用处：让后备字体贴住主字体的基线。单位可以用任何 TEX 长度单位，`em` 随字号一起变化。两条声明给出同一个字体时，各自保留自己的值；一个字体实例只记一个位置，所以一段文字取排出它时所选的字族的那个值。

```
\setCJKfamilyfont{high}[Baseline = .3em]
  {FandolKai-Regular.otf}
\songti 基线\quad\CJKfamily{high}基线
```

基线 基线

### 3.4 加载 Fandol

`LoadFandol=`⟨**true**|*false*⟩ 默认打开。当文档没有声明任何 CJK 字体时，在 `\begin{document}` 处加载四款 Fandol 字体：宋体作为主字体，黑体作为无衬线字体，仿宋作为等宽字体，并且把宋、黑、楷、仿宋全部注册为具名字

族。宋体一族以楷体作为斜体，所以 \emph 与 \itshape 会把汉字改成楷体。已经声明了字体的文档保留自己的字体，所以 Fandol 不会混进来；但若它恰好没有声明 \CJKfamilydefault 所用的那一族，正文就会用 \CJKrmdefault 那一族，若也没有则用最先声明的一族，并且给出警告。什么字体都没声明又关掉这个键时，同样会给出警告。

### 3.5 伪粗体与伪斜体

一种 CJK 字族常常没有粗体或斜体。下面这些键让 LuaCJK 请求 fontspec 加粗或倾斜常规字体，而不去使用粗体或斜体。它们作用于此后注册的字族，并且在字族自己设置了 BoldFont 或 ItalicFont 时不生效。它们是全局的：分组不会改变一个已经注册的字族所用的设置。一个既没有设置 ItalicFont 也没有打开 AutoFakeSlant 的字族没有斜体形状：强调的文字保持正体，fontspec 会报告缺少这个形状。

AutoFakeBold = ⟨*true*|**false**|*number*⟩ 在请求粗体形状时加粗常规字体。给出一个数字时同时设置 EmboldenFactor。

AutoFakeSlant = ⟨*true*|**false**|*number*⟩ 在请求斜体形状时倾斜常规字体。给出一个数字时同时设置 SlantFactor。

EmboldenFactor = ⟨*number*⟩: 加粗的量。初始值是 4。

SlantFactor = ⟨*number*⟩: 倾斜的量，介于 −0.999 与 0.999 之间。初始值是 0.167。

```
\LuaCJKsetup{AutoFakeBold = true}
\setCJKmainfont{FandolSong-Regular.otf}
\textbf{English 中文字体}\\
English 中文字体
```

**English 中文字体**
English 中文字体

\defaultCJKfontfeatures `\defaultCJKfontfeatures {⟨font features⟩}`

设置前置到此后注册的每一个 CJK 字族及其子区块与后备字体的特性。初始值是 Script = CJK。在调用之前注册的字族保留它们当时得到的特性。

\addCJKfontfeatures `\addCJKfontfeatures [⟨ignored⟩] {⟨font features⟩}`

为当前 CJK 字族添加特性，直到分组结束，例如 \addCJKfontfeatures{Scale = 1.2}。开头的一个 * 也可以接受；两种形式作用于同一个字体。

```
\defaultCJKfontfeatures{Scale = 1.5}
\setCJKfamilyfont{big}
  {FandolSong-Regular.otf}
\CJKfamily{big}中文字体\\
\CJKfamily{zhsong}中文字体\\
\addCJKfontfeatures{Scale = 1.2}
\CJKfamily{zhsong}中文字体
```

中文字体
中文字体
中文字体

### 3.6 数学模式中的汉字

CJKmath = ⟨*true*|**false**⟩: 是否允许在数学模式里直接输入汉字。打开时，这样的字符用它自己的一个数学字族的 CJK 字体排出，而不是到数学字体里去找，所以上标或分数里的字符会取那个数学样式的尺寸。可以用 `\LuaCJKsetup{CJKmath = true}` 打开它，也可以用宏包选项 `CJKmath = true`。此时字符取数学版本（见 `\mathversion`）的字体：在 `bold` 版本里用字族的粗体，字族没有粗体时保持正体；字形取字族的正体，不随周围的正文。不打开它时，数学模式里的汉字会到数学字体里去找，通常就此消失，所以这种情况下 LuaCJK 给出一次性的警告。

`\setCJKmathfont` `\setCJKmathfont [⟨options⟩] {⟨font⟩} [⟨options⟩]`

选择数学模式里的字符所用的 CJK 字体。不用这个命令时，使用 CJK 主字体。本手册在前导区打开 `CJKmath` 并设置了这个字体。与其他的字体命令一样，这个选择只作用于当前分组：分组里的公式使用该分组里选择的字族。

数学模式里出现的每一个 CJK 字族都被注册成一个独立的数学字族，这样 TEX 就能取它的文本、脚本或脚脚本尺寸。字族按数学版本选字体：`bold` 版本用 `\bfdefault`，其余版本用 `\mddefault`，字形的形状取 `\shapedefault`。所以 `\boldmath` 会把公式里的汉字换成粗体，而且只作用于当前分组；字族没有粗体时保持正体，也不报警告。字族号取自 240–255 这一段，是 LuaTEX 允许的 256 个字族里的最后十六个，正常文档不会用到那里；数学里用到的每一个后备字体也各占一个号，声明了它的字族共用这个号，一个文档如果需要十六个以上的号就会报错停止。脚本尺寸是当前正文尺寸的百分之七十五，也就是标准尺寸表里的比例。这个正文尺寸是公式周围文字的尺寸，而不是公式里 `\text` 临时切换到的尺寸，所以含有 `\text` 的公式与不含它的公式一样大。数学参数不取自 CJK 字体，因为它没有 `MathConstants` 表：这些参数仍然是文档数学字体的，这就是为什么两个汉字组成的分数用的是周围数学的横线与间距。

```
$ 天地玄黄 $ and 天地玄黄
```

$ 天地玄黄 $ and 天地玄黄

# 4 文本排版

## 4.1 间距

CJKglue = ⟨*glue*⟩: 插入相邻 CJK 字符之间的胶。初始值是 `0pt plus 0.08\baselineskip`，在每次选择字体时重新计算，所以它跟随字号；文档给出的值按原样使用，不随字号变化，连同它的伸缩阶（例如 `plus 1fill`）一起保留。

CJKecglue = ⟨*glue*⟩: 插入 CJK 与拉丁字符之间的胶。标点两侧的胶没有宽度：标点的进距本身就带有间距。半角的 ASCII 括号只留下开头括号之前的这一份胶，括号内侧不加，于是成对的括号贴住它括起来的文字。初始值是 `0.3333em plus 0.1667em minus 0.1111em`，它同样跟随字号。

CJKspace = ⟨*true*|**false**⟩: 是否保留源文件中 CJK 字符之间的空格。不保留时，空格换成字符之间的胶，断行与伸缩都随之保留。用 `\ ` 或 `\nobreakspace` 写出的空格不是源文件中的空格，它的宽度就是一个空格的宽度。

xCJKecglue = ⟨*true*|**false**⟩: 是否把源文件中 CJK 与拉丁字符之间的空格替换为 `CJKecglue`。

| | |
|---|---|
| `中文 English 中文` | 中文 English 中文 |
| `\LuaCJKsetup{CJKspace = true}天 地 玄 黄 宇 宙` | 天 地 玄 黄 宇 宙 |

`\normalspacedchars` `\normalspacedchars {⟨character list⟩}`

列出的字符两侧都不加胶，这样拉丁文字中的连字符或斜线不会把周围的文字拉开。初始列表是 `/`、`\` 与 `-`。这个设置是全局的。

| | |
|---|---|
| `中x中\par`<br>`\normalspacedchars{x}中x中` | 中 x 中<br>中x中 |

只用于控制文字、本身没有进距的字符分两类。U+200B、U+2060、U+180E 与 U+FEFF 不含字体能用的信息，在输入时就已删除，因此它们既不占胶，也不把两个 CJK 字符分开。连接符（U+200C、U+200D）与各个变体选择符留给字体：它们跟随前一个字符的字体，所以一个 emoji 序列不会被拆成两个图形，一个变体序列也能选到它的字形；字体用不到它们时留下的占位字形在断行之前就去掉，因此它们既不占进距，也不在两个 CJK 字符之间加上 `CJKecglue`。用 `\char` 请求的字符不会被删除，因为 `\char` 要的就是字形本身。盒子同样按它占的进距计：不占进距的盒子不把两侧的文字分开，所以 `\rlap`、`\llap` 与 `\clap` 覆盖在文字上的内容仍属于同一串。

CJKfilltwosides (*env.*) 把各行排进一个竖直盒子里，盒中的行两端对齐到 ⟨*width*⟩，各行之间用 \\ 分开。它的 ⟨*position*⟩ 参数像 \parbox 一样决定盒子落在基线上的方式： t 让第一行的基线与周围的行齐平， c 表示居中（默认）， b 让最后一行的基线与周围的行齐平。环境像 \parbox 一样先离开竖直模式，所以它后面的文字接着盒子排在同一行，而不是另起一段；环境之后的空格与 xeCJK 一样被忽略。宽度为零或为负时沿用周围的行宽。带星号的形式行为相同；它不与内容的自然宽度作比较。

```
\begin{CJKfilltwosides}[⟨position⟩]{⟨width⟩}
⟨lines⟩
\end{CJKfilltwosides}
```

```
\begin{CJKfilltwosides}{6em}
  瞻彼阕者，\\虚室生白
\end{CJKfilltwosides}
```

瞻彼阕者，
虚室生白

```
\begin{CJKfilltwosides*}{0pt}
  瞻彼阕者，\\虚室生白
\end{CJKfilltwosides*}
```

瞻 彼 阕 者 ，
虚 室 生 白

## 4.2 标点

PunctStyle = ⟨*style*⟩: 选择一种标点样式。已经实现的样式有 quanjiao（默认）、 plain、banjiao、kaiming、CCT 与 hangmobanjiao。未知的名字给出警告并回退到 plain。

PunctWidth = ⟨*length*|*false*⟩: 不管样式怎么说，给每个标点同样的进距。 0pt 也是一个进距，标点因此不占任何进距； false（初始值）让标点回到样式给出的宽度。用 \LuaCJKsetwidth 为某个标点设置的宽度仍然优先。

PunctBoundWidth = ⟨*length*|*false*⟩: 标点出现在行尾或行首时，它的进距中有多少留在行内；其余的墨迹悬挂到页边，或者悬挂到段落开头或列表项开头的缩进里。默认悬挂标点自带的空白，所以它的墨迹正好落在版心边上；断行时这一份悬挂已经算进去，行尾的标点因此能随它所在的行一起放下。居中的段落与右对齐的段落没有页边可悬挂，所以它的开头与 \\ 强制换行之后的一行都不悬挂，断行处的悬挂照旧，因为断行时它已经算进了这一行。 0pt 把整个进距都悬挂出去， false 恢复这个默认值。

LatinPunct = ⟨*true*|**false**⟩: 把引号（U+2018、U+2019、U+201C、U+201D）、间隔号（U+00B7）、两点前导符（U+2025）、省略号（U+2026、U+22EF）与连字点

(U+2027) 当作拉丁标点：它们留在当前的拉丁字体里，也不被压缩。这样英文单词里的撇号就不会变成一个全角形式。破折号不受影响。

PunctFamily = ⟨*family*|*false*⟩: 为标点使用另一个 CJK 字族。该字族必须用 \setCJKfamilyfont 声明；false （初始值）让标点重新跟随文字。该字族的 Baseline 同样移动它排出的标点。

AllowBreakBetweenPuncts = ⟨*true*|**false**⟩: 两个相邻的标点是否可以被断行分开。破折号与省略号不在此列，一串连续的破折号或省略号由 NoBreakLongPunct 控制。收尾的、中间的标点与破折号、省略号不能出现在行首，开头的标点不能出现在行尾，这两条禁则不由本选项放开，所以一个收尾的引号不会被单独排在行首。标点仍然会被压缩。

PunctKinsoku = ⟨**true**|*false*⟩: 是否对标点施加断行禁则。收尾的、中间的标点与破折号、省略号不能出现在行首，开头的标点不能出现在行尾，半角的 ASCII 括号同样如此：开头的不能出现在行尾，收尾的不能出现在行首；两个相邻的标点不会被分开，一串连续的破折号或省略号保持在一起。盒子边缘的标点同样参与判断。这些禁则与 PunctStyle 无关，所以 PunctStyle = plain 时它们仍然有效。

```
不亦说乎，不亦乐乎\\
\LuaCJKsetup{PunctFamily = zhkai}
不亦说乎，不亦乐乎
```

不亦说乎，不亦乐乎
不亦说乎，不亦乐乎

```
\LuaCJKsetup{LatinPunct = false}%
a’a\quad 不亦说乎！\\
\LuaCJKsetup{LatinPunct, PunctWidth = .8em}
a’a\quad 不亦说乎！
```

a’ a 不亦说乎！
a′a 不亦说乎！

```
\fbox{甲，乙}\\
\LuaCJKsetup{PunctWidth = 0pt}%
\fbox{甲，乙}
```

甲，乙
甲乙

```
列位看官。\\
\LuaCJKsetup{PunctBoundWidth = .5em}
列位看官。\\
（列位看官
```

列位看官。
列位看官。
（列位看官

接下来是标点代码所用的字符集合。每个集合的键都有三种形式：不带符号的形式替换整个集合，带 + 的形式添加字符，带 - 的形式去掉字符。范围的写法与 \LuaCJKDeclareCharClass 相同。这些集合是全局的：分组不会把先前的成员带回来。

KaiMingPunct<+->=⟨*char list*⟩: 在 kaiming 样式里保持全角宽度的标点。初始为。、．、？、！。

MiddlePunct<+->=⟨*char list*⟩: 居中的标点，放在自己盒子的两半之间。居中的标点不能出现在行首。

LongPunct<+->=⟨*char list*⟩: 彼此之间从不压缩的标点，例如破折号与省略号。

NoBreakLongPunct<+->=⟨*char list*⟩: 不能被断行分开的长标点。初始与 LongPunct 相同。

---

\LuaCJKDeclareCharClass `\LuaCJKDeclareCharClass {⟨class⟩} {⟨range list⟩}`

改写一段码位的类别。范围用十六进制书写，前导的 " 可以省略，"START -> "END 表示一段范围；列表用逗号分隔。类别有：CJK 与 Default 让这段范围表现得像 CJK 或拉丁文字； FullLeft 与 HalfLeft 把它当作开头的标点， FullRight 与 HalfRight 把它当作收尾的标点，于是断行禁则适用；前者是全角标点，照样压缩与悬挂，后者只能是 ASCII 括号：它不压缩、不悬挂，只在成对时贴住括号内的文字。 NormalSpace 把这段范围移出 CJK/拉丁胶，就像 \normalspacedchars 那样； CJStarter、CM 与 Boundary 当作 CJK 处理，因为 LuaTEX 没有组合标记或文字段边界的模型。

---

\LuaCJKDeclarePunctClass `\LuaCJKDeclarePunctClass {⟨group⟩} {⟨range list⟩}`

改写一段范围的标点组。组有 open、close、 middle、dash 与 ellipsis，以及半角括号的两组 half-open 与 half-close； none 去掉标点处理，于是这些字符不再被压缩或悬挂。

---

\LuaCJKResetCharClass 丢弃迄今声明的每一个类别与标点改写，并把正常间距集合恢复为 \normalspacedchars 的字符。

```
中ABC中\par
\LuaCJKDeclareCharClass{CJK}{0041 -> 005A}
中ABC中\par
\LuaCJKDeclarePunctClass{none}{FF0C}%
字，，字\par
\LuaCJKResetCharClass 中ABC中
```

中 ABC 中
中ABC中
字，，字
中 ABC 中

---

\LuaCJKEditPunctStyle `\LuaCJKEditPunctStyle {⟨style⟩} {⟨key-value list⟩}`

修改一个已有样式的参数。键是 compress-adjacent, adjacent-ratio, keep-open-close；没有给出的键保持它原来的值。

\LuaCJKsetwidth `\LuaCJKsetwidth {⟨punctuation list⟩} {⟨length⟩}`

给列出的标点一个固定的进距，覆盖所选的 PunctStyle 与 PunctWidth。这个设置是全局的：分组不会恢复先前的宽度，后一次调用是在表中添加，而不是替换。开头的一个 * 可以接受；它会在行首或行尾设置的宽度没有实现，星号只警告一次。

\LuaCJKsetkern `\LuaCJKsetkern {⟨before⟩} {⟨after⟩} {⟨length⟩}`

设置两个相邻标点之间的空隙，覆盖样式的设置。每个参数只使用第一个字符。与 \LuaCJKsetwidth 一样，这个设置是全局的。

```
\LuaCJKsetwidth{「}{1.5em}字「字\\
\LuaCJKsetkern{「}{」}{.3em}字「」字
```

字 「字
字 「」字

\LuaCJKDeclarePunctStyle `\LuaCJKDeclarePunctStyle {⟨name⟩} {⟨key-value list⟩}`

定义一种标点样式。

compress-adjacent = ⟨**true**|*false*⟩: 是否压缩相邻的标点。

adjacent-ratio = ⟨*number*⟩: 压缩的比例，相对于被压缩标点的宽度。

keep-open-close = ⟨**true**|*false*⟩: 开标点与闭标点相邻时是否保持不压缩。

```
\LuaCJKDeclarePunctStyle{mine}
  {adjacent-ratio = .25}
字，，字\par
\LuaCJKsetup{PunctStyle = mine}字，，字\par
\LuaCJKEditPunctStyle{mine}
  {adjacent-ratio = .75}
字，，字
```

字，，字
字，，字
字，，字

### 4.3 断行

一串破折号或省略号（U+2013、U+2014、U+2015、U+FF0D、U+2026 与 U+22EF）是一个整体：这些标点之间不加胶，所以两端对齐不会把一个两字宽的破折号拆开，它们之间也不提供断行机会。这一串标点之前也不提供断行机会，所以一行不会以破折号或省略号开头。键 LongPunct 替换这个集合。

两个字符之间显式的 \nobreak 仍是断行禁止，它们之间的胶也照常插入：胶紧跟在惩罚之后，它提供的断行机会因而仍被禁止。若惩罚之前有一个源文件里的空格，这个空格也换成字符之间的胶，惩罚随之移到胶的前面。

基线被移动的盒子依附于它前面的字符：\textsuperscript 与 \textsubscript 就是这样排的。这种盒子两侧都不加胶，它前面也不提供断行机会，所以它的内容不会另起一行。

CheckSingle = ⟨*true*|**false**⟩ 打开时，末字符是 CJK 的段落的最后两个断行点被设为不可断开，这样单个字符就不会孤零零地落在最后一行上。初始值是 ⟨*false*⟩。

WidowPenalty = ⟨*penalty*⟩: CheckSingle 使用的惩罚值。初始值是 10000；较小的值在值得这个代价时允许断开。

CJLineBreak = ⟨*normal*|*strict*⟩: 在 strict 模式下，Unicode 的 Line_Break 属性为 CJ 的字符（小假名之类）不能出现在行首。这个模式只增加禁则，不取消任何禁则。初始值是 normal，它把这些字符当作其他 CJK 字符。这个集合从 unicode-data 宏包集的 LineBreak.txt 读取，所以严格模式需要这个宏包集。

```
\parbox{3em}{天地玄黄宇宙洪荒\par}\quad
\LuaCJKsetup{CheckSingle = true}%
\parbox{3em}{天地玄黄宇宙洪荒\par}
```

天地玄　天地玄
黄宇宙　黄　宇
洪荒　　宙洪荒

\LuaCJKnobreak　\LuaCJKnobreak {⟨*text*⟩}

把 ⟨*text*⟩ 保持在一行之内。参数的节点被打上标记，断行的处理不在它们的任何边界处断开，参数里由 \quad 或 \hspace 写出的胶也一样。标记随当前分组结束。

```
\parbox{3em}{天地玄黄}\quad
\parbox{3em}{\LuaCJKnobreak{天地玄黄}}
```

天地玄
黄　天地玄黄

## 4.4 暂停与钩子

\LuaCJKdisable
\LuaCJKenable

\LuaCJKdisable 停止排版调整：不再插入 CJKglue 或 CJKecglue，标点不再压缩，不再添加断行禁则，标点不再悬挂，数学模式里的 CJK 字符也不再处理。CJK 字符仍然用 CJK 字体排出，所以文字依然可读。这个开关只作用于当前分组；\LuaCJKenable 重新打开这些调整。

这就是抄录类环境使用的钩子，在测量原始节点列表时也用得上。

```
精致甲版 红楼梦\\
\LuaCJKdisable 精致甲版 红楼梦
```

精致甲版红楼梦
精致甲版 红楼梦

\LuaCJKShipoutHook `\LuaCJKShipoutHook {⟨code⟩}`

在每一页输出之前运行 ⟨*code*⟩。代码被加入 `shipout/before` 钩子，所以它遵守通常的钩子顺序。

## 4.5 抄录与 listings

\LuaCJKVerbAddon
\LuaCJKOffVerbAddon

让一个 CJK 字符和当前等宽字体的两个空格一样宽，这样代码列表里含有 CJK 文字时各列仍然对齐。空隙以 `CJKglue` 的形式加入，所以 CJK 字体本身并没有被缩放；负的空隙只给出警告，什么也不加。 CJK 字符之间的断行被禁止，直到分组结束，与拉丁文的抄录文字一样。

这条命令在最外层的分组里不起作用，所以它应当在需要它的环境内部使用，例如 fancyvrb 的 `formatcom` 选项里。它在自己的分组里重新打开排版调整：要补的间距只有经过调整才会加进去，所以在 `Verb` 为 `true` 或 `env` 的抄录环境里它同样有效。 `\LuaCJKOffVerbAddon` 在同一个分组里再次取消它，并恢复它打开调整之前暂停的状态。

```
\ttfamily abc汉字abc\\
\ttfamily\LuaCJKVerbAddon
abc汉字abc
```

```
abc汉字abc
abc汉字abc
```

`Verb` = ⟨**true**|*false*|*env*|*env+*⟩ `\verb` 与抄录类环境（`verbatim`、`Verbatim`、`lstlisting` 以及它们的近亲）里做什么。`true` 在两者中都暂停调整； `env` 在环境里保留调整，但在 `\verb` 里不保留，这样代码里含有 CJK 文字时各列仍然对齐； `env+` 在所有地方都保留调整；`false` 完全不碰抄录材料。既然 `env+` 与 `false` 都保留调整，它们在 LuaCJK 里行为相同。暂停只去掉间距，不取折断行机会：被暂停材料两边的零宽胶仍在，所以紧挨着一个 CJK 字符的 `\verb` 可以断行，紧挨着被暂停材料的一个标点也保留它后面的断行机会。无论这个设置取什么值，`\LuaCJKVerbAddon` 都会在它所在的分组里把调整重新打开。 fancyvrb 的行内命令 `\Verb` 与 `\UseVerb` 按 `\verb` 的规则处理；块命令 `\VerbatimInput` 与 `\UseVerbatim` 则与抄录环境一致。

listings 一次处理一个字符，并且在读到字符的那一刻就把字符表之外的字符排出来。CJK 字符在它那张表之外，于是会跑到周围拉丁文字的前面。配套的 LuaCJK-listings 修正了这一点：它把 CJK 字符加入 listings 的字符表，所以它们留在原处，把每个字符算作两列，并在列表内部关掉间距调整。一串 CJK 字符对 `emph` 与 `keywords` 选项来说是一个词，所以可以给一个 CJK 变量名着色；拉丁字母、数字和 CJK 标点都会结束这个词。使用 listings 时 LuaCJK 自动加载它，所以不需要额外的命令。

\lstinline 以改变了的类别码读取参数；补丁会破坏那个扫描器，所以行内代码保持不动，按通常的方式处理。

### 4.6 装饰

配套的 LuaCJKfntef 提供下划线、着重号与删除线。它建立在 lua-ul 之上，所以被装饰的文字可以包含任意命令、CJK 与拉丁文字，也仍然可以被断字并跨行断开。

\CJKunderline `\CJKunderline [⟨options⟩] {⟨text⟩}`

给 ⟨*text*⟩ 加下划线。初始几何是 `depth = 0.2em` 与 `thickness = 0.4pt`。开头的一个 `*` 或 `-` 可以接受，并且被忽略。

所有装饰命令共用下面这组键：与某个命令无关的键照样接受，只是不起作用。只给一个颜色名而不写键名，等价于 `format =` ⟨*color*⟩。键 `top` 与 `bottom` 只对 `\CJKunderline` 有效，并覆盖计算出来的几何，不论横线是否由 `format` 着色。画出的图案从不改变被装饰行的高度或深度，所以行距不会变大。下划线、双线、波浪、斜删除线与符号都按当前字号各建一个图案，所以字体变大时图案跟着变大。

`thickness` = ⟨*length*⟩: 横线的粗细，初始 `0.4pt`。

`depth` = ⟨*length*⟩: 从基线到横线顶部的距离，初始 `0.2em`；着重号的点是它在 em 盒底部之下的深度。

`textformat` = ⟨*code*⟩: 作用于文字的代码，例如 `textformat = \bfseries`。

`format` = ⟨*color name*⟩: 横线的颜色，例如 `format = red`；它只影响横线，不影响文字。

`color` = ⟨*color name*⟩: `format` 的另一个名字。

`subtract` = ⟨*length*⟩: 在横线两端各去掉的长度，初始关闭；`true` 表示半字宽，这样相邻的两个命令不会连在一起。

`skip` = ⟨*true*|**false**⟩: 把标点排除在横线之外。

`hidden` = ⟨*true*|**false**⟩: 隐藏文字而保留横线，横线取隐藏文字的宽度、高度与深度；文字照常断行，也不留在 PDF 里，所以读不到、选不中。

`gap` = ⟨*length*⟩: 双线两条横线之间的距离，初始 `1.1pt`。

`height` = ⟨*length*⟩: 删除线中心到基线的距离，初始 `0.3em`。

`symbol` = ⟨*code*⟩: 替代波浪或圆点的符号。

selectable = ⟨*true*|**false**⟩: 装饰符号是否保留为文字，初始 false；true 时 PDF 阅读器可以复制与选择它。

不给 format 时，横线使用它所装饰的文字的颜色，例如 \textcolor 设定的颜色。

sep 为了兼容而接受，但没有作用。

```
\CJKunderline[thickness = 1pt]{有涯}\\
\CJKunderline[format = red]{有涯}\\
\CJKunderline[textformat = \bfseries]
  {有涯}\\
\CJKunderline[hidden]{有涯}
```

有涯
有涯
**有涯**

```
今天是 \CJKunderline{\hspace*{3em}} 年\\
第 \CJKunderline{\hspace*{2em}} 题
```

今天是 _____ 年
第 ____ 题

```
\CJKunderline{有涯}\CJKunderline{有涯}\\
\CJKunderline[subtract]{吾生也有涯}\\
\CJKunderline[skip]{有涯，无涯}
```

有涯有涯
吾生也有涯
有涯，无涯

\CJKunderdblline \CJKunderdblline [⟨*options*⟩] {⟨*text*⟩}

在 ⟨*text*⟩ 下面画两条平行的横线，键见上面的键列表。两条横线都使用 thickness，上面那条从 depth 开始， gap 是两条横线之间的距离。开头的一个 * 或 - 可以接受，并且被忽略。

\CJKunderwave \CJKunderwave [⟨*options*⟩] {⟨*text*⟩}

在 ⟨*text*⟩ 下面画一条波浪线。波浪线用 l3draw 画出，各个周期首尾相接：一个周期宽 1em，上下各 .25em。一段整字的文字由整数个周期组成，波浪线从第一个字的边缘画到最后一个字的边缘；被对齐拉伸时，整段按最终宽度分成整数个周期，周期跟着整行的拉伸变长或变短。整个图案随字号缩放；与其他画出的图案一样，它不改变行的高度与深度。 symbol 用来把波浪线换成别的符号：每字宽一个符号，居中，符号顶部在基线之下半字宽处。

```
\CJKunderwave{吾生也有涯}\\
\CJKunderwave[symbol = \textbullet]
  {吾生也有涯}\\
\CJKunderdot{吾生也有涯}\\
\CJKunderdot[symbol = \textbullet]
  {吾生也有涯}
```

吾生也有涯
吾生也有涯
• • • • •
吾生也有涯
吾生也有涯
• • • • •

\CJKxout `\CJKxout [⟨options⟩] {⟨text⟩}`

用重复的斜线划掉 ⟨*text*⟩，像 ulem 的 `\xout` 那样。斜线从基线之上 `0.8em` 延伸到基线之下 `-0.15em`，所以它穿过一个 CJK 字形的整个字身。

\LuaCJKemdescent 当前字体 em 盒的深度，可以当作长度使用。CJK 的 em 盒是一个 em 的正方形，所以把高度为 `1em` 的对象放在这个深度上，就正好覆盖一个汉字所占的区域；字体没有 OpenType 度量时退回 `.12em`。

\CJKunderanyline `\CJKunderanyline {⟨depth⟩} {⟨object⟩} {⟨text⟩}`

把 ⟨*object*⟩ 放在 ⟨*text*⟩ 下面，并用 `\leaders` 重复。⟨*object*⟩ 的底部放在基线之下 ⟨*depth*⟩ 处，所以对于一个（本身没有深度的）`\rule` 来说，横线从基线之下 ⟨*depth*⟩ 处向上占据。要覆盖一个 CJK 字形，就用字体的 em 盒：⟨*depth*⟩ 取 `\LuaCJKemdescent`，对象高度取 `1em`，正如下例所做的那样。对象填满的是 em 盒，也就是 PDF 阅读器划出的选区：横向是字身宽度，纵向是 em 盒的高度，字体墨迹比 em 盒窄时，四周会各露出一条边距。类型会被缓存，所以相同的参数复用一个下划线类型。

\CJKunderanysymbol `\CJKunderanysymbol[⟨options⟩]{⟨depth⟩}{⟨symbol⟩}{⟨text⟩}`

用法相同，只是每字宽放一个 ⟨*symbol*⟩：符号在一个和一个 CJK 字符一样宽的格子里居中，所以一排符号与它们上方的字符对齐。⟨*depth*⟩ 是符号顶部到*em* 盒底部的距离，所以 `0pt` 留下的间隙正与字符两侧的边距相同；字体没有 OpenType 度量时按 `.12em` 计算。为了兼容，接受选项 `sep`，但它没有作用，因为格子固定为一个字宽。

```
\CJKunderanyline{\LuaCJKemdescent}
  {\color{yellow}\rule{1pt}{1em}}{有涯}\\
\CJKunderanysymbol{0pt}{\tiny$\triangle$}{有涯}
```

有涯
有涯
△ △

\CJKunderdot `\CJKunderdot [⟨options⟩] {⟨text⟩}`

在 ⟨*text*⟩ 下面加着重号。每个 CJK 字符下面一个点，点在字符的一字宽方格内居中，顶部在 em 盒底部之下 `0.2em` 处；拉丁字母、数字和空格下面没有点，一段文字断行以后，每一行的字符各有自己的点，粘连伸长时点随字符移动。用 `symbol =` ⟨*symbol*⟩ 时，一个符号取代着重号，放置方式与 `\CJKunderanysymbol` 的符号相同：每字宽一个，居中，符号顶部在 em 盒底部之下 `0.3em` 处。

```
\CJKunderline{虚室生白，\CJKunderdot{吉祥}止止，
  虚室生白}
```

虚室生白，吉祥止止，
虚室生白

\CJKsout `\CJKsout [⟨options⟩] {⟨text⟩}`

给 ⟨*text*⟩ 加删除线，键见上面的键列表。删除线中心在基线上方，穿过一个 CJK 字形的中部，比拉丁字母 x 高度的中部要高；对于完全是拉丁文的文字，可以用 `height = 0.5ex`。

```
\CJKsout[thickness = 2pt, format = red]
  {吾生也有涯}\\
\CJKsout[height = .5ex]{吾生也有涯}
```

~~吾生也有涯~~
~~吾生也有涯~~

```
\CJKunderline   {吾生也有涯}\\
\CJKunderdblline{吾生也有涯}\\
\CJKunderwave   {吾生也有涯}\\
\CJKunderdot    {吾生也有涯}\\
\CJKsout        {吾生也有涯}\\
\CJKxout        {吾生也有涯}
```

吾生也有涯
吾生也有涯
吾生也有涯
吾生也有涯
吾生也有涯
吾生也有涯

# 5 与 xeCJK 的比较

LuaCJK 只要两个引擎允许，就沿用 xeCJK 的接口，所以一个已经在用 xeCJK 的文档大多不需要改动。这里只列出不同之处。

## 5.1 行为差异

- Verb：env+ 的行为与 false 一样，因为两者都保留调整。
- 标点几何。样式把标点的尺寸给成一半，或者一半但句末标点保持全角，或者 0.7 倍，并且墨迹按类别随进距移动：开头的标点移动整个量，中间的标点移动一半，收尾的标点完全不移动。 xeCJK 则是测量每个字形的真实侧边距。
- \LuaCJKDeclarePunctStyle 接受三个键 compress-adjacent、adjacent-ratio 与 keep-open-close；xeCJK 的三十个几何键描述它的侧边距模型，在这里没有对应物。
- CJKfilltwosides* 不与自然宽度作比较。
- CJK 字体在文字定形之前就应用到字符上，所以 CJK 字体的 OpenType 特性生效：组合记号附到它的基字上，变体序列选到它的字形，字体自己的连字（例如两个破折号并成一个两字宽的破折号）与 locl 形式也会出现；间距、标点进距与禁则仍然来自 LuaCJK 自己。

- 在 LuaCJKfntef 里，`format` 接受颜色名而不是代码，而且装饰由 lua-ul 画出，所以 `ulem` 的选项列表不存在，`\underLine` 直接配置。

## 5.2 未提供的功能

- `NewLineCS`、`EnvCS`、`InlineEnv` 与 `PlainEquation`：它们的存在是为了让 xeCJK 在扫描记号时认出 TEX 构造，而一个已经完成的节点列表不需要提示。
- `PoZheHaoLigature`：字体自带的两字宽破折号连字在定形时就会出现，所以没有开关它的选项。
- `RubberPunctSkip`：标点的进距是一个 kern，它既不是断行机会，也不参与两端对齐，而正是这一点让固定宽度的样式保持稳定。
- `CheckFullRight`：它需要标点之后的控制序列，节点列表并不显示这个；请改用 `\LuaCJKnobreak`。

# 1 Introduction

LuaCJK is a LuaTEX-only package for Chinese typesetting. Its main interface is `\setCJKmainfont`, `\setCJKsansfont`, `\setCJKmonofont`, `\setCJKfamilyfont`, `\CJKfamily`, and `\LuaCJKsetup`.

The current CJK font is recorded in a LuaTEX attribute, and each CJK glyph is set in that font before the text is shaped.

# 2 Loading

```
\usepackage[⟨options⟩]{LuaCJK}
```

LuaCJK requires the LuaTEX engine. It conflicts with luatexja: loading either package before or after the other raises an error.

`\LuaCJKversion` Print the version of LuaCJK.

`\LuaCJKsetup` `\LuaCJKsetup {⟨key-value list⟩}`

Configure LuaCJK. The keys are the ones documented in the following sections.

The keys fall into two groups.

The settings that describe how text is set are local to the current group, and they are recorded on the characters themselves, so text inside a group keeps its settings even when the group has already ended by the time the paragraph is broken:

```
\vbox{\begingroup\LuaCJKsetup{CJKglue = 10pt}中中\endgroup\par}
```

uses the 10pt glue, although the group is closed before `\par`. This group contains

- `CJKglue`, `CJKecglue`, `CJKspace`, and `xCJKecglue`;
- `PunctStyle`, `PunctKinsoku`, `AllowBreakBetweenPuncts`, `PunctWidth`, and `PunctBoundWidth`;
- `LatinPunct` and `CJKmath`;
- `CheckSingle`, `WidowPenalty`, and `CJLineBreak`;
- `AutoFallBack` and `Verb`;
- the two commands `\LuaCJKCancelSubCJKBlock` and `\LuaCJKRestoreSubCJKBlock`.

The other keys are global or act while the package is loaded:

- `KaiMingPunct`, `MiddlePunct`, `LongPunct`, and `NoBreakLongPunct` edit character sets;
- `AutoFakeBold`, `AutoFakeSlant`, `EmboldenFactor`, and `SlantFactor` act when a family is registered;
- `LocalConfig` acts while the package is loaded, and `LoadFandol` at `\begin{document}`.

The declarations `\LuaCJKDeclareCharClass`, `\LuaCJKDeclarePunctClass`, `\LuaCJKDeclareSubCJKBlock`, `\LuaCJKsetwidth`, and `\LuaCJKsetkern` are global as well, as noted with each of them.

The same keys may be given as package options:

```
\usepackage[PunctStyle = banjiao]{LuaCJK}
```

`LocalConfig` = ⟨**true**|*false*|*name*⟩ Read local settings from a configuration file. `true` loads `luacjk.cfg`, ⟨*name*⟩ loads `luacjk-`⟨*name*⟩`.cfg`, and `false` (the initial value) loads nothing. The file is read at the end of the package, so it may use every command LuaCJK defines. A missing file only warns.

To use LuaCJK with ctex, pass the `engine` option with the value ⟨*luacjk*⟩:

```
\documentclass[engine = luacjk]{ctexart}
```

ctex then inputs LuaCJK's `ctex-engine-luacjk.def` instead of its own LuaTEX backend, and that file loads the package, so LuaCJK is not loaded separately.

LuaCJK cooperates with the LATEX tagged-PDF project. A document that declares

```
\DocumentMetadata{tagging = on}
```

marks the ink of a decoration as `artifact`, and a decoration symbol is left out of the text a reader copies or selects; `selectable = true` keeps the symbol as text. Under the l3doc class, load latex-lab-testphase-l3doc as well; this manual is built that way. A PDF reader derives the height of a selection from the font's `hhea` ascent and descent, which is 1.437em for `Noto Serif CJK SC`; the selection is therefore taller than the 1em square. LuaCJK cannot change that, and it does not affect the glyph.

## 3 Fonts

The main font commands take an optional list of fontspec options. Options may be written before or after the font name. A CJK font is always selected in the Unicode encoding, whatever encoding the Latin text uses, so fontenc's `T1` option leaves the CJK text alone.

`\setCJKmainfont` `\setCJKsansfont` `\setCJKmonofont`

`\setCJKmainfont [⟨options⟩] {⟨font⟩} [⟨options⟩]`

Set the CJK font for the roman, sans serif, and monospaced families.

```
\setCJKmainfont[Scale = 1.2]
  {FandolSong-Regular.otf}
English 中文字体\\
\setCJKsansfont[Scale = 1.2]
  {FandolKai-Regular.otf}
\sffamily English 中文字体
```

English 中文字体
English 中文字体

`\setCJKfamilyfont` `\setCJKfamilyfont{⟨family⟩}[⟨options⟩]{⟨font⟩}[⟨options⟩]`

Define a named CJK family. The family may be given as `\CJKrmdefault` and friends.

`\CJKfamily` `\CJKfamily {⟨family⟩}`

Switch to a named CJK family. The family may be given as `\CJKrmdefault` and friends.

```
\CJKfamily{zhsong}English 中文字体\\
\CJKfamily{zhhei}English 中文字体\\
\CJKfamily{zhkai}English 中文字体\\
\CJKfamily{zhfs}English 中文字体
```

English 中文字体
English 中文字体
English 中文字体
English 中文字体

`\newCJKfontfamily` `\newCJKfontfamily[⟨family⟩]⟨command⟩ [⟨options⟩]{⟨font⟩}[⟨options⟩]`

Define a command that switches to a newly declared family. The command is local to the current group, like `\newfontfamily`. The options may be given before or after the font name.

\CJKfontspec `\CJKfontspec [⟨options⟩] {⟨font⟩}`

Select a CJK font for the current group. The options go before the font name, as for `\fontspec`: the command is used inside the text, so if it also read a bracket after the font name, the bracket that the text begins with would be taken as an option list and would vanish.

`\songti`
`\heiti`
`\fangsong`
`\kaishu`
`\lishu`
`\youyuan`

Shortcuts for the families named `zhsong`, `zhhei`, `zhfs`, `zhkai`, `zhli`, and `zhyou`.

```
\newCJKfontfamily[kai]\mykai
  {FandolKai-Regular.otf}
  \mykai English 中文字体\\
  \songti English 中文字体
```

English 中文字体
English 中文字体

`\CJKrmdefault`
`\CJKsfdefault`
`\CJKttdefault`
`\CJKfamilydefault`

The CJK families selected by `\rmfamily`, `\sffamily`, and `\ttfamily`, and the family selected by `\normalfont`. `\CJKfamilydefault` follows `\familydefault` by default: it is the CJK family that corresponds to the main family when that is one of the three, and `\CJKrmdefault` otherwise. It is selected when the document starts, so redefining it or `\familydefault` in the preamble sets the main text in that family from the first line.

### 3.1 Sub-blocks

One CJK font rarely covers every block of Unicode. A sub-block gives a range of characters its own font inside a family; the block then becomes an option of the font commands.

\LuaCJKDeclareSubCJKBlock `\LuaCJKDeclareSubCJKBlock {⟨block⟩} {⟨range list⟩}`

Declare ⟨*block*⟩ with the given ranges, in the same spelling as `\LuaCJKDeclareCharClass`. The name becomes a font option, as the example below shows: a family that sets it uses the second font for the characters of the block, and a family that does not leaves them in the main font. Overlapping blocks are resolved in declaration order.

```
\LuaCJKDeclareSubCJKBlock{Kana}{3040 -> 30FF}
\setCJKmainfont[Kana = FandolKai-Regular.otf]
  {FandolSong-Regular.otf}
汉字 かんじ
```

汉字かんじ

\LuaCJKCancelSubCJKBlock `\LuaCJKCancelSubCJKBlock {⟨block list⟩}`
\LuaCJKRestoreSubCJKBlock `\LuaCJKRestoreSubCJKBlock {⟨block list⟩}`

Turn the listed blocks off or on again for the rest of the group. Because the choice of font is recorded per font identifier, two families that supply the same font share one setting: the family that selected the font last decides.

### 3.2 Fallback fonts

A CJK font rarely covers every character one needs. Fallback fonts are tried in the order they were declared when the selected CJK font has no glyph for a character.

\setCJKfallbackfamilyfont `\setCJKfallbackfamilyfont {⟨family⟩} [⟨options⟩] {⟨font⟩} [⟨options⟩]`

Add ⟨*font*⟩ to the fallback list of ⟨*family*⟩. The family may be given as `\CJKrmdefault` and friends. Call the command several times to build a chain; the first font that has the character wins. The chain is resolved in the weight, shape, and size of the text, so a character in bold takes the bold face of the fallback font. With `CJKmath` on, a CJK character in math mode falls back through the same chain: the list of the CJK family that supplies the math font is tried, at the size of the math style.

`AutoFallBack` = ⟨*true*|**false**⟩ Whether to use the declared fallback fonts. The switch follows the text like the spacing keys, so it can be turned on for one group.

```
\setCJKmainfont{FandolSong-Regular.otf}
\setCJKfallbackfamilyfont{\CJKrmdefault}
  {FandolKai-Regular.otf}
\LuaCJKsetup{AutoFallBack = true}中あ
```

中あ

### 3.3 Baseline shift

Two CJK fonts are rarely designed for each other: mixed in one line, the characters of one font sit too high or too low. A font may carry a `Baseline` value that moves its characters up or down, so that the two fonts stand on one baseline. The value belongs to the font rather than to the text, which is why it is an option of the command that declares the font.

`Baseline` = ⟨*dimension*⟩ Raise the characters of this font by ⟨*dimension*⟩; a negative value lowers them. It is an option like the others, so a fallback declaration may carry it as well, and that is its usual use: it makes a fallback font stand on the baseline of the main font. Any TEX unit works, and `em` follows the size. Two declarations that supply the same font keep their own values; a font instance carries one position, so a paragraph takes the value of the family that is selected when it is set.

```
\setCJKfamilyfont{high}[Baseline = .3em]
  {FandolKai-Regular.otf}
\songti 基线\quad\CJKfamily{high}基线
```

基线　基线

### 3.4 Loading Fandol

`LoadFandol` = ⟨**true**|*false*⟩ On by default. When the document declares no CJK font, load the four Fandol faces at `\begin{document}`: Song as the main font, Hei as the sans serif font, Fang as the monospaced font, and all of Song, Hei, Kai, and Fang as the named families. The Song family takes Kai as its italic face, so `\emph` and `\itshape` set the text in Kai. A document that declares fonts keeps its own, so Fandol does not mix into it; when it does not declare the family `\CJKfamilydefault` names, its text takes the family `\CJKrmdefault` names, or the first declared family when that one is missing too, and a warning says so. A document that declares nothing with the key off is warned about as well.

### 3.5 Fake bold and slant

A CJK family often has no bold or italic face. The following keys make LuaCJK ask fontspec to embolden or slant the regular face instead. They apply to families registered afterwards, and are not used when the family itself sets `BoldFont` or `ItalicFont`. They are global: a group does not change what an already registered family uses. A family that names neither `ItalicFont` nor `AutoFakeSlant` has no italic shape: emphasized text keeps its upright face, and fontspec reports the shape as missing.

AutoFakeBold = ⟨*true*|**false**|*number*⟩ Embolden the regular face when the bold shape is requested. A number also sets `EmboldenFactor`.

AutoFakeSlant = ⟨*true*|**false**|*number*⟩ Slant the regular face when the italic shape is requested. A number also sets `SlantFactor`.

EmboldenFactor = ⟨*number*⟩: The amount of emboldening. The initial value is `4`.

SlantFactor = ⟨*number*⟩: The amount of slanting, between −0.999 and 0.999. The initial value is `0.167`.

```
\LuaCJKsetup{AutoFakeBold = true}
\setCJKmainfont{FandolSong-Regular.otf}
\textbf{English 中文字体}\\
English 中文字体
```

**English 中文字体**
English 中文字体

`\defaultCJKfontfeatures` `\defaultCJKfontfeatures {⟨font features⟩}`

Set the features prepended to every CJK family registered afterwards, and to the font of each of its sub-blocks and fallbacks. The initial value is `Script = CJK`. Families registered before the call keep the features they were given.

`\addCJKfontfeatures` `\addCJKfontfeatures [⟨ignored⟩] {⟨font features⟩}`

Add features to the current CJK family for the rest of the group, for example `\addCJKfontfeatures{Scale = 1.2}`. A leading `*` is accepted; both forms act on the same font.

```
\defaultCJKfontfeatures{Scale = 1.5}
\setCJKfamilyfont{big}
  {FandolSong-Regular.otf}
\CJKfamily{big}中文字体\\
\CJKfamily{zhsong}中文字体\\
\addCJKfontfeatures{Scale = 1.2}
\CJKfamily{zhsong}中文字体
```

中文字体
中文字体
中文字体

### 3.6 CJK in mathematics

CJKmath = ⟨*true*|**false**⟩: Whether CJK characters may be typed directly in math mode. When it is on, such a character is set in the CJK font of a math family of its own instead of being looked up in the math font, so a character in a superscript or a fraction takes the size of that math style. Enable it with

`\LuaCJKsetup{CJKmath = true}` or with the package option `CJKmath = true`. Such a character takes the font of the math version, as `\mathversion` sets it: the bold face of the family in the `bold` version, its regular face when it has no bold one, and the upright shape of the family rather than the shape of the text. Without it a CJK character in math mode is looked up in the math font and usually disappears, so LuaCJK issues a one-time warning in that case.

`\setCJKmathfont` `\setCJKmathfont [⟨options⟩] {⟨font⟩} [⟨options⟩]`

Select the CJK font used for characters in math mode. Without this command the CJK main font is used. This manual enables `CJKmath` and sets this font in its preamble. The selection is local to the current group, like the other font commands: a formula in a group uses the family selected in that group.

Each CJK family that appears in math is registered as a math family of its own, so that TEX can pick the text, script, or scriptscript size of it. The family takes its font from the math version: `\bfdefault` in the `bold` version and `\mddefault` in the others, with the shape `\shapedefault`. `\boldmath` therefore bolds the CJK characters of a formula, within the group that asks for it, and a family without a bold face keeps its regular one without a warning. The family number comes from the range 240–255, the last sixteen of the 256 that LATEX $2_\varepsilon$ allocates under LuaTEX, which an ordinary document never reaches; every fallback font used in math takes a number of its own, shared by the families that declare it, and a document that needs more than sixteen numbers stops with an error. The script sizes are seven and five tenths of the current text size, the ratio of the standard size tables. That text size is the one of the text around the formula, not the size a `\text` inside it switches to, so a formula that contains a `\text` is as large as one that does not. The math parameters are not taken from the CJK font, which has no `MathConstants` table: they stay those of the document's math font, which is why a fraction of two CJK characters has the rule and the gaps of the surrounding math.

| | |
|---|---|
| `$ 天地玄黄 $ and 天地玄黄` | $ 天地玄黄 $ and 天地玄黄 |

# 4 Text layout

## 4.1 Spacing

`CJKglue` = ⟨*glue*⟩: Glue inserted between adjacent CJK characters. The initial value is `0pt plus 0.08\baselineskip`, recomputed at every font selection, so it follows the

size; a value given by the document is used as written and does not change with the size, and its stretch order (for example `plus 1filll`) is kept as well.

`CJKecglue` = ⟨*glue*⟩: Glue inserted between CJK and Latin characters. Beside a punctuation mark the glue has no advance: the mark's own advance already carries the spacing. A half-width ASCII bracket keeps that glue only in front of an opening one; the inside of the pair takes none, so the brackets hug the text they enclose. The initial value is `0.3333em plus 0.1667em minus 0.1111em`, which follows the size in the same way.

`CJKspace` = ⟨*true*|**false**⟩: Whether to keep source spaces between CJK characters. When they are not kept, the space becomes the inter-character glue, so the break opportunity and the stretch survive. A space written with `\ ` or `\nobreakspace` is not a source space, and its advance is that of a space.

`xCJKecglue` = ⟨*true*|**false**⟩: Whether to replace source spaces between CJK and Latin characters with `CJKecglue`.

| | |
|---|---|
| `中文 English 中文` | 中文 English 中文 |
| `\LuaCJKsetup{CJKspace = true}天 地 玄 黄 宇 宙` | 天 地 玄 黄 宇 宙 |

`\normalspacedchars` `\normalspacedchars {`⟨*character list*⟩`}`

The listed characters get no glue on either side, so that a hyphen or a slash inside Latin text does not pull the surrounding text apart. The initial list is `/`, `\`, and `-`. The setting is global.

| | |
|---|---|
| `中x中\par`<br>`\normalspacedchars{x}中x中` | 中x中<br>中x中 |

Characters that only control the text and have no advance of their own come in two kinds. U+200B, U+2060, U+180E, and U+FEFF carry nothing a font can use, so they are removed at the input: they neither take glue nor separate two CJK characters. The joiners (U+200C, U+200D) and the variation selectors are left to the font: each follows the font of the character in front of it, so an emoji sequence is not split into two pictures and a variation sequence selects its glyph; what the font leaves of them is removed before the line is broken, so they take no advance and add no `CJKecglue` between two CJK characters. A character requested with `\char` is not removed, because `\char` asks for the glyph itself. A box is read by the advance it occupies in the same way: one that occupies none does not separate

the text around it, so what `\rlap`, `\llap` and `\clap` set over the text stays in one run.

`CJKfilltwosides` (*env.*) Set the lines in a vertical box whose lines are justified to ⟨*width*⟩, separated by `\\`. Its ⟨*position*⟩ argument places the box as `\parbox` does: `t` puts the baseline of the first line on the surrounding baseline, `c` centres the box on the text axis (the default), and `b` puts the last line there. The environment leaves vertical mode first, as `\parbox` does, so the text after it joins the box on the same line instead of beginning a paragraph, and a space after the environment is ignored as xeCJK ignores it. A width of zero or less keeps the surrounding line width. The starred form behaves the same; it does not compare against the natural width of the content.

```
\begin{CJKfilltwosides}[⟨position⟩]{⟨width⟩}
⟨lines⟩
\end{CJKfilltwosides}
```

```
\begin{CJKfilltwosides}{6em}
  瞻彼闋者，\\虚室生白
\end{CJKfilltwosides}
```

瞻彼闋者，
虚室生白

```
\begin{CJKfilltwosides*}{0pt}
  瞻彼闋者，\\虚室生白
\end{CJKfilltwosides*}
```

瞻 彼 闋 者 ，
虚 室 生 白

## 4.2 Punctuation

`PunctStyle` = ⟨*style*⟩: Select a punctuation style. The implemented styles are `quanjiao` (the default), `plain`, `banjiao`, `kaiming`, `CCT`, and `hangmobanjiao`. An unknown name issues a warning and falls back to `plain`.

`PunctWidth` = ⟨*length*|*false*⟩: Give every punctuation mark the same advance, whatever the style says. `0pt` is an advance too, so the marks take none at all; `false` (the initial value) returns them to the width the style gives. A width set with `\LuaCJKsetwidth` still wins for that mark.

`PunctBoundWidth` = ⟨*length*|*false*⟩: How much of a mark's advance stays inside the line when the mark ends or starts the line; the rest of the ink stands in the margin, or in the indentation at the start of a paragraph or of a list item. The default leaves the mark's own blank out, so its ink stands on the measure, and line breaking counts that hang, so a mark can end the line the natural width would refuse. A centered

or a flush-right paragraph has no margin to hang into, so the paragraph start and a line after \\ do not hang there; the hang at a break stays, because line breaking has already counted it. `0pt` hangs the whole advance; `false` returns to that default.

`LatinPunct` = ⟨*true*|**false**⟩: Treat the quotation marks (U+2018, U+2019, U+201C, U+201D), the middle dot (U+00B7), the two dot leader (U+2025), the ellipses (U+2026, U+22EF), and the hyphenation point (U+2027) as Latin punctuation: they stay in the current Latin font and are not compressed. This keeps an apostrophe inside an English word from turning into a full-width form. The em dash is not affected.

`PunctFamily` = ⟨*family*|*false*⟩: Use a different CJK family for the punctuation marks. The family must be declared with `\setCJKfamilyfont`; `false` (the initial value) makes the marks follow the text again. The `Baseline` of that family moves the marks it sets as well.

`AllowBreakBetweenPuncts` = ⟨*true*|**false**⟩: Whether two adjacent punctuation marks may be separated by a line break. A dash or an ellipsis is not affected: a run of them is governed by `NoBreakLongPunct`. The prohibitions that keep a closing, middle, dash, or ellipsis mark out of the line start, and an opening mark out of the line end, are not lifted, so a closing quotation mark is never set at the start of a line. The marks are still compressed.

`PunctKinsoku` = ⟨**true**|*false*⟩: Whether to apply line-breaking prohibitions for punctuation. A closing, middle, dash, or ellipsis punctuation cannot start a line, an opening punctuation cannot end a line, and the same holds for a half-width ASCII bracket: an opening one cannot end a line and a closing one cannot start one; two adjacent marks are not separated, and a run of long punctuation is kept together. A mark at the edge of a box takes part as well. The prohibitions are independent of `PunctStyle`, so they stay active with `PunctStyle = plain`.

```
不亦说乎，不亦乐乎\\
\LuaCJKsetup{PunctFamily = zhkai}
不亦说乎，不亦乐乎
```

不亦说乎，不亦乐乎
不亦说乎，不亦乐乎

```
\LuaCJKsetup{LatinPunct = false}%
a’a\quad 不亦说乎！\\
\LuaCJKsetup{LatinPunct, PunctWidth = .8em}
a’a\quad 不亦说乎！
```

a’a　不亦说乎！
a’a　不亦说乎！

```
\fbox{甲，乙}\\
\LuaCJKsetup{PunctWidth = 0pt}%
\fbox{甲，乙}
```

甲，乙
甲乙

```
列位看官。\\
\LuaCJKsetup{PunctBoundWidth = .5em}
列位看官。\\
（列位看官
```

列位看官。
列位看官。
（列位看官

The character sets that the punctuation code works with come next. Each of their keys has three forms: the plain one replaces the set, the form with + adds characters, and the form with - removes them. Ranges use the same spelling as `\LuaCJKDeclareCharClass`. The sets are global: a group does not bring the previous members back.

`KaiMingPunct<+->` = ⟨*char list*⟩: The marks that keep a full width in the `kaiming` style. Initially 。、．、？、！.

`MiddlePunct<+->` = ⟨*char list*⟩: The marks centered between the two halves of their box. A middle mark cannot start a line.

`LongPunct<+->` = ⟨*char list*⟩: The marks that are never compressed against each other, such as dashes and ellipses.

`NoBreakLongPunct<+->` = ⟨*char list*⟩: The long marks that must not be separated by a line break. Initially the same as `LongPunct`.

`\LuaCJKDeclareCharClass` `\LuaCJKDeclareCharClass {⟨class⟩} {⟨range list⟩}`

Override the class of a code-point range. Ranges are written in hexadecimal and the leading `"` may be omitted; `"START -> "END` gives a range, and the list is comma separated.

The classes are: `CJK` and `Default` make the range behave like CJK or Latin text; `FullLeft` and `HalfLeft` treat it as an opening mark, `FullRight` and `HalfRight` as a closing mark, so the line-breaking prohibitions apply; the first of each pair is a full-width mark and is compressed and hung as well, while the second is an ASCII bracket: it keeps its own advance, does not hang, and hugs the text between the two of a pair. `NormalSpace` takes the range out of the CJK/Latin glue, like `\normalspacedchars`; `CJStarter`, `CM`, and `Boundary` are treated as `CJK`, because LuaTEX has no model of combining marks or run boundaries.

\LuaCJKDeclarePunctClass `\LuaCJKDeclarePunctClass {⟨group⟩} {⟨range list⟩}`

Override the punctuation group of a range. The groups are `open`, `close`, `middle`, `dash`, and `ellipsis`, and `half-open` and `half-close` for the half-width brackets; `none` removes the punctuation treatment, so the characters are no longer compressed or hung.

\LuaCJKResetCharClass Drop every class and punctuation override declared so far, and put the normal-spaced set back to the characters of `\normalspacedchars`.

```
中ABC中\par
\LuaCJKDeclareCharClass{CJK}{0041 -> 005A}
中ABC中\par
\LuaCJKDeclarePunctClass{none}{FF0C}%
字，，字\par
\LuaCJKResetCharClass 中ABC中
```

中 ABC 中
中ABC中
字，，字
中 ABC 中

\LuaCJKEditPunctStyle `\LuaCJKEditPunctStyle {⟨style⟩} {⟨key-value list⟩}`

Change the parameters of an existing style. The keys are `compress-adjacent`, `adjacent-ratio`, `keep-open-close`; a key that is not given keeps its value.

\LuaCJKsetwidth `\LuaCJKsetwidth {⟨punctuation list⟩} {⟨length⟩}`

Give the listed punctuation marks a fixed advance, overriding the selected `PunctStyle` and `PunctWidth`. The setting is global: a group does not restore the previous width, and a later call adds to the table rather than replacing it. A leading `*` is accepted; the width it would set at the start or end of a line is not implemented, and the star warns once.

\LuaCJKsetkern `\LuaCJKsetkern {⟨before⟩} {⟨after⟩} {⟨length⟩}`

Set the gap between two adjacent punctuation marks, overriding the style. Only the first character of each argument is used. Like `\LuaCJKsetwidth`, the setting is global.

```
\LuaCJKsetwidth{「}{1.5em}字「字\\
\LuaCJKsetkern{「}{」}{.3em}字「」字
```

字 「字
字 「」字

\LuaCJKDeclarePunctStyle `\LuaCJKDeclarePunctStyle {⟨name⟩} {⟨key-value list⟩}`

Define a punctuation style.

`compress-adjacent` = ⟨**true**|*false*⟩: whether two adjacent punctuation marks are compressed.

`adjacent-ratio` = ⟨*number*⟩: the ratio of the compression, relative to the width of the mark that is compressed.

`keep-open-close` = ⟨**true**|*false*⟩: whether an opening mark next to a closing one stays uncompressed.

```
\LuaCJKDeclarePunctStyle{mine}
  {adjacent-ratio = .25}
字，，字\par
\LuaCJKsetup{PunctStyle = mine}字，，字\par
\LuaCJKEditPunctStyle{mine}
  {adjacent-ratio = .75}
字，，字
```

字，，字
字，，字
字，，字

### 4.3 Line breaking

A run of dashes or ellipses (U+2013, U+2014, U+2015, U+FF0D, U+2026, and U+22EF) is one unit: the marks are joined without glue, so justification cannot pull a two-em dash apart, and no break is offered between them. No break is offered in front of the run either, so a line does not begin with a dash or an ellipsis. The key `LongPunct` replaces the set.

An explicit `\nobreak` between two characters stays a prohibition, and the glue between them is still inserted: the glue follows the penalty, so the break it offers is still forbidden. A source space in front of the penalty becomes that glue as well, and the penalty moves in front of it.

A box whose baseline has been moved belongs to the character in front of it: `\textsuperscript` and `\textsubscript` set their content in one. No glue is inserted beside such a box, and it offers no break before it, so its content never starts a line of its own.

`CheckSingle` = ⟨*true*|**false**⟩ When on, the last two break points of a paragraph whose final character is CJK are made unbreakable, so a single character cannot end up alone on the last line. The initial value is ⟨*false*⟩.

`WidowPenalty` = ⟨*penalty*⟩: The penalty used by `CheckSingle`. The initial value is `10000`; a lower value allows the break when it is worth the cost.

CJLineBreak = ⟨*normal*|*strict*⟩: In the `strict` mode a character whose Unicode `Line_Break` property is `CJ` (the small kana and the like) may not start a line. The mode only adds prohibitions; it does not remove any. The initial value is `normal`, which treats those characters like other CJK characters. The set is read from `LineBreak.txt` of the unicode-data bundle, so the package needs that bundle for the strict mode.

```
\parbox{3em}{天地玄黄宇宙洪荒\par}\quad
\LuaCJKsetup{CheckSingle = true}%
\parbox{3em}{天地玄黄宇宙洪荒\par}
```

天地玄　天地玄
黄宇宙　黄　宇
洪荒　　宙洪荒

`\LuaCJKnobreak` `\LuaCJKnobreak {⟨text⟩}`

Keep ⟨*text*⟩ on one line. The nodes of the argument are marked and the line-breaking pass forbids a break at any of their boundaries, including a glue the argument writes with `\quad` or `\hspace`. The marking ends with the current group.

```
\parbox{3em}{天地玄黄}\quad
\parbox{3em}{\LuaCJKnobreak{天地玄黄}}
```

天地玄
黄　天地玄黄

## 4.4 Suspending and hooks

`\LuaCJKdisable` `\LuaCJKenable`

`\LuaCJKdisable` stops the typographic adjustments: no CJKglue or CJKecglue is inserted, punctuation is not compressed, no line-breaking prohibitions are added, punctuation is not hung, and CJK characters in math mode are left alone. CJK characters are still typeset in the CJK font, so text stays readable. The switch is local to the current group; `\LuaCJKenable` turns the adjustments back on.

This is the hook verbatim-like environments use, and it is also useful when measuring raw node lists.

```
精致甲版 红楼梦\\
\LuaCJKdisable 精致甲版 红楼梦
```

精致甲版红楼梦
精致甲版 红楼梦

`\LuaCJKShipoutHook` `\LuaCJKShipoutHook {⟨code⟩}`

Run ⟨*code*⟩ before every page is shipped out. The code is added to the `shipout/before` hook, so it obeys the usual hook ordering.

### 4.5 Verbatim and listings

\LuaCJKVerbAddon
\LuaCJKOffVerbAddon

Make one CJK character as wide as two spaces of the current monospaced font, which keeps columns aligned when a code listing contains CJK text. The gap is added as `CJKglue`, so the CJK font itself is not scaled; a negative gap only raises a warning and adds nothing. Breaks between CJK characters are forbidden until the group ends, as in Latin verbatim text.

The command does nothing at the outermost group level, so it is meant to be used inside the environment that needs it, for example in the `formatcom` option of fancyvrb. It turns the typographic adjustments back on inside its own group, because the gap it computes reaches the text only through them, so it works in a verbatim-like environment whose `Verb` setting is `true` or `env` as well. `\LuaCJKOffVerbAddon` cancels it again inside the same group and restores the suspension that was in force before the addon turned it off.

```
\ttfamily abc汉字abc\\
\ttfamily\LuaCJKVerbAddon
abc汉字abc
```

abc汉字abc
abc汉字abc

`Verb` = ⟨**true** | *false* | *env* | *env+*⟩ What to do inside `\verb` and the verbatim-like environments (`verbatim`, `Verbatim`, `lstlisting`, and their relatives). `true` suspends the adjustments in both; `env` keeps them inside the environments but not in `\verb`, which preserves column alignment when the code contains CJK text; `env+` keeps them everywhere; `false` leaves verbatim material entirely alone. Since `env+` and `false` both leave the adjustments on, they behave the same in LuaCJK. Suspension removes the spacing only, not the break: the zero-width glue at either edge of suspended material stays, so a `\verb` beside a CJK character can end a line, and a punctuation mark next to suspended material still keeps the break opportunity that follows it. Whatever the setting is, `\LuaCJKVerbAddon` turns the adjustments back on inside the group it is used in. The inline `\Verb` and `\UseVerb` of fancyvrb are treated as `\verb` is, and its block commands `\VerbatimInput` and `\UseVerbatim` as the environments are.

listings processes one character at a time and typesets a character that is outside its character table at the moment it is read. CJK characters are outside that table, so they end up in front of the surrounding Latin text. The companion package LuaCJK-listings fixes this: it adds the CJK characters to the character table of listings, so they stay in place, counts each of them as two columns, and turns the spacing adjustments off inside the listing. A run of CJK characters is one word

for the `emph` and `keywords` options, so a CJK variable name can be highlighted; a Latin character, a digit, or a CJK mark ends the word. LuaCJK loads it automatically when listings is used, so no extra command is needed.

`\lstinline` reads its argument with changed category codes; the patch would break that scanner, so inline code is left alone and takes the normal handling.

### 4.6 Decorations

The companion package LuaCJKfntef provides underlining, emphasis dots, and strike-through. It builds on lua-ul, so the decorated text may contain arbitrary commands, CJK and Latin text, and may still be hyphenated and broken between lines.

`\CJKunderline` `\CJKunderline [⟨options⟩] {⟨text⟩}`

Underline ⟨*text*⟩. The initial geometry is `depth = 0.2em` and `thickness = 0.4pt`. A leading `*` or `-` is accepted and ignored.

Every decoration takes the keys below; a key that a command does not use is accepted and has no effect. A bare color name with no key means `format =` ⟨*color*⟩. The keys `top` and `bottom` apply to `\CJKunderline` alone and override the computed geometry, whether or not `format` colors the rule. A drawn pattern never changes the height or depth of the line it decorates, so the leading does not grow. The underline, the double rule, the wave, the slash, and a symbol each get one pattern per font size, so a pattern grows with the size of the text.

`thickness` = ⟨*length*⟩: the thickness of the rule, initially `0.4pt`.

`depth` = ⟨*length*⟩: the distance from the baseline to the top of the rule, initially `0.2em`; for the emphasis dots it is their depth below the bottom of the em box.

`textformat` = ⟨*code*⟩: code applied to the text, for example `textformat = \bfseries`.

`format` = ⟨*color name*⟩: the color of the rule, for example `format = red`; it affects the rule only, not the text.

`color` = ⟨*color name*⟩: another name for `format`.

`subtract` = ⟨*length*⟩: taken off the rule at both ends, off by default; `true` means half an em, so two neighbouring commands do not run together.

`skip` = ⟨*true*|**false**⟩: leaves the punctuation out of the rule.

`hidden` = ⟨*true*|**false**⟩: hides the text and keeps the rule, with the width, height, and depth of the hidden text; the text still breaks between lines, and no glyph of it reaches the PDF, so it cannot be read or selected.

`gap` = ⟨*length*⟩: the distance between the two rules of the double line, initially `1.1pt`.

`height` = ⟨*length*⟩: the distance from the baseline to the center of the strike-out rule, initially `0.3em`.

`symbol` = ⟨*code*⟩: a symbol replacing the wave or the dots.

`selectable` = ⟨*true*|**false**⟩: whether the decoration symbol stays as text, initially `false`; with `true` a PDF reader can copy and select it.

Without `format` the rule takes the color of the text it decorates, for example the color set by `\textcolor`.

`sep` is accepted for compatibility and has no effect.

```
\CJKunderline[thickness = 1pt]{有涯}\\
\CJKunderline[format = red]{有涯}\\
\CJKunderline[textformat = \bfseries]
  {有涯}\\
\CJKunderline[hidden]{有涯}
```

有涯
有涯
**有涯**

```
今天是 \CJKunderline{\hspace*{3em}} 年\\
第 \CJKunderline{\hspace*{2em}} 题
```

今天是 _____ 年
第 ____ 题

```
\CJKunderline{有涯}\CJKunderline{有涯}\\
\CJKunderline[subtract]{吾生也有涯}\\
\CJKunderline[skip]{有涯，无涯}
```

有涯有涯
吾生也有涯
有涯，无涯

---

`\CJKunderdblline` `\CJKunderdblline [⟨options⟩] {⟨text⟩}`

Put two parallel rules under ⟨*text*⟩, with the keys listed above. Both rules use `thickness`, the top rule starts at `depth`, and `gap` is the distance between them. A leading `*` or `-` is accepted and ignored.

\CJKunderwave `\CJKunderwave [⟨options⟩] {⟨text⟩}`

Put a wave under ⟨*text*⟩. The wave is drawn with l3draw, and its periods stay joined: one period is `1em` wide and reaches `.25em` above and below. A span of whole characters is an exact number of periods, so the wave runs from the first character's edge to the last one's; when justification stretches the span, the span is divided into whole periods at its final width, so the period follows the stretch of the line. Like the other drawn patterns it leaves the height and depth of the line alone. `symbol` replaces the wave with another symbol: one symbol per em, centered, its top half an em below the baseline.

```
\CJKunderwave{吾生也有涯}\\
\CJKunderwave[symbol = \textbullet]
  {吾生也有涯}\\
\CJKunderdot{吾生也有涯}\\
\CJKunderdot[symbol = \textbullet]
  {吾生也有涯}
```

吾生也有涯
吾生也有涯
吾生也有涯
吾生也有涯
吾生也有涯

\CJKxout `\CJKxout [⟨options⟩] {⟨text⟩}`

Strike ⟨*text*⟩ out with a repeated slash, like ulem's `\xout`. The slash runs from `-0.15em` to `0.8em` relative to the baseline, so it crosses the whole body of a CJK glyph.

\LuaCJKemdescent The depth of the em box of the current font, usable as a length. A CJK em box is one em square, so an object of height `1em` placed at this depth covers the area a CJK character occupies; a font without OpenType metrics falls back to `.12em`.

\CJKunderanyline `\CJKunderanyline {⟨depth⟩} {⟨object⟩} {⟨text⟩}`

Put ⟨*object*⟩ under ⟨*text*⟩, repeated by `\leaders`. The bottom of ⟨*object*⟩ is placed ⟨*depth*⟩ below the baseline, so for a `\rule` (which has no depth of its own) the rule occupies ⟨*depth*⟩ below the baseline upwards. To cover a CJK character, use the em box of the font: ⟨*depth*⟩ is `\LuaCJKemdescent` and the object is `1em` high, as the example below does. The object fills the em box, which is the area a PDF reader selects: the advance across, the em box down, so a font whose ink is smaller shows its side bearing all round. The type is cached, so the same arguments reuse one underline type.

\CJKunderanysymbol `\CJKunderanysymbol[⟨options⟩]{⟨depth⟩}{⟨symbol⟩}{⟨text⟩}`

The same, with one ⟨*symbol*⟩ per em: the symbol is centered in a cell as wide as a CJK character, so a row of symbols lines up with the characters above them. ⟨*depth*⟩ is the distance from the top of the symbol to the *bottom of the em box*, so `0pt` leaves the same gap below the character as the side bearings leave beside it; a font without OpenType metrics is computed at `.12em`. The option `sep` is accepted for compatibility but has no effect, because the cell is fixed at one em.

```
\CJKunderanyline{\LuaCJKemdescent}
  {\color{yellow}\rule{1pt}{1em}}{有涯}\\
\CJKunderanysymbol{0pt}{\tiny$\triangle$}{有涯}
```

有涯
有涯
△ △

\CJKunderdot `\CJKunderdot [⟨options⟩] {⟨text⟩}`

Put emphasis dots under ⟨*text*⟩. Each CJK character gets one dot, centred in the one-em cell of the character, its top `0.2em` below the bottom of the em box; Latin letters, digits, and spaces get none, each line of a broken span carries its own dots, and the dots follow the characters when the inter-character glue stretches. With `symbol = ⟨symbol⟩` a symbol takes the place of the dots and is placed like the symbol of `\CJKunderanysymbol`: one per em, centered, its top `0.3em` under the bottom of the em box.

```
\CJKunderline{虚室生白，\CJKunderdot{吉祥}止止，
  虚室生白}
```

虚室生白，吉祥止止，
虚室生白

\CJKsout `\CJKsout [⟨options⟩] {⟨text⟩}`

Strike ⟨*text*⟩ through, with the keys listed above. The initial height passes through the middle of a CJK glyph, which sits higher than the middle of the Latin x-height; use `height = 0.5ex` for text that is entirely Latin.

```
\CJKsout[thickness = 2pt, format = red]
  {吾生也有涯}\\
\CJKsout[height = .5ex]{吾生也有涯}
```

~~吾生也有涯~~
~~吾生也有涯~~

```
\CJKunderline   {吾生也有涯}\\
\CJKunderdblline{吾生也有涯}\\
\CJKunderwave   {吾生也有涯}\\
\CJKunderdot    {吾生也有涯}\\
\CJKsout        {吾生也有涯}\\
\CJKxout        {吾生也有涯}
```

吾生也有涯
吾生也有涯
吾生也有涯
吾生也有涯
~~吾生也有涯~~
~~吾生也有涯~~

# 5 Comparison with xeCJK

LuaCJK follows the xeCJK interface where the two engines allow it, so a document that already uses xeCJK mostly needs no change. Only the differences are listed.

## 5.1 Different behavior

- `Verb`: `env+` behaves like `false`, because both leave the adjustments on.
- Punctuation geometry. A style gives a mark half the size, half with the sentence-ending marks kept full, or 0.7 of it, and the ink moves with the advance by class: an opening mark by the whole amount, a middle mark by half, a closing mark not at all. xeCJK measures the real side bearings of each glyph instead.
- `\LuaCJKDeclarePunctStyle` takes the three keys `compress-adjacent`, `adjacent-ratio`, and `keep-open-close`; the thirty geometry keys of xeCJK describe its side-bearing model and have no counterpart here.
- `CJKfilltwosides*` does not compare with the natural width.
- The CJK font is applied to the characters before the text is shaped, so the OpenType features of the CJK font take effect: a combining mark attaches to its base, a variation sequence selects its glyph, and the font's own ligatures and `locl` forms appear, two dashes joined into one two-em dash for instance. The spacing, the punctuation advance, and the prohibitions still come from LuaCJK itself.
- In LuaCJKfntef, `format` takes a color name rather than code, and the decoration is drawn by lua-ul, so the `ulem` option list does not exist and `\underLine` is configured directly.

## 5.2 Not available

- `NewLineCS`, `EnvCS`, `InlineEnv`, and `PlainEquation`: they exist so that xeCJK can recognize TEX constructs while it scans tokens, and a finished node list needs no hint.
- `PoZheHaoLigature`: the font's own two-em dash ligature applies when it has one, so there is no option that switches it on.

- `RubberPunctSkip`: the advance of a mark is a kern, which is neither a break opportunity nor part of justification, and that is what keeps a fixed-width style stable.
- `CheckFullRight`: it needs the control sequence after the mark, which a node list does not show; use `\LuaCJKnobreak`.

# A Implementation

```
1 ⟨*package⟩
2 ⟨@@=LuaCJK⟩
```

## A.1 Loading and conflicts

LuaCJK only works with LuaTEX. The engine test runs before any other package is loaded, so that a wrong engine stops with a clear message.

```
3 \sys_if_engine_luatex:F
4   {
5     \msg_new:nnn { LuaCJK } { engine }
6       { LuaCJK ~ requires ~ LuaTeX. }
7     \msg_critical:nn { LuaCJK } { engine }
8     \tex_endinput:D
9   }
10 \RequirePackage { fontspec }
```

\LuaCJKversion The version of the package. The version string, taken from the package metadata.

```
11 \tl_const:Nx \c__LuaCJK_version_tl { \ExplFileVersion }
12 \cs_new:Npn \LuaCJKversion { \tl_use:N \c__LuaCJK_version_tl }
```

(*End of definition for* `\LuaCJKversion` *and* `\c__LuaCJK_version_tl`*. This function is documented on page 23.*)

luatexja installs its own CJK font and spacing layer and works on the same callbacks as LuaCJK, so the two are rejected whatever the loading order. xeCJK needs no check here: it stops by itself on any engine other than XETEX.

```
13 \msg_new:nnn { LuaCJK } { luatexja-conflict }
14   {
15     LuaCJK ~ and ~ LuaTeX-ja ~ must ~ not ~ be ~ loaded ~ together.\\
16     Load ~ only ~ one ~ of ~ them.
17   }
18 \cs_if_exist:cT { ver@luatexja.sty }
19   { \msg_critical:nn { LuaCJK } { luatexja-conflict } }
20 \hook_gput_code:nnn { package / luatexja / before } { LuaCJK }
21   { \msg_critical:nn { LuaCJK } { luatexja-conflict } }
```

## A.2 Registers, styles, and options

The runtime state lives in attributes and registers, which the options below fill in.

\g__LuaCJK_font_attr
\c__LuaCJK_font_attr_int
\g__LuaCJK_cjk_space_attr
\c__LuaCJK_cjk_space_attr_int
\g__LuaCJK_xecglue_attr
\c__LuaCJK_xecglue_attr_int
\g__LuaCJK_inserted_attr
\c__LuaCJK_inserted_attr_int
\g__LuaCJK_math_font_attr
\c__LuaCJK_math_font_attr_int
\g__LuaCJK_disabled_attr
\c__LuaCJK_disabled_attr_int
\g__LuaCJK_punct_font_attr
\c__LuaCJK_punct_font_attr_int
\g__LuaCJK_settings_attr
\c__LuaCJK_settings_attr_int
\g__LuaCJK_nobreak_attr
\c__LuaCJK_nobreak_attr_int
\g__LuaCJK_attached_attr
\c__LuaCJK_attached_attr_int
\g__LuaCJK_explicit_space_attr
\c__LuaCJK_explicit_space_attr_int

The attributes record, per node, the CJK font, the space and glue policies, whether the node was inserted by LuaCJK, the CJK math font, whether the adjustments are suspended, whether the character is one that attaches to the character in front of it, and whether a space is one the document wrote with `\ ` or `\nobreakspace`. The integer constants hold the allocated attribute numbers for the Lua side.

```
\clist_map_inline:nn
  {
    font,     cjk_space,  xecglue,   inserted,   math_font,
    disabled, punct_font, settings,  nobreak,    attached,
    explicit_space
  }
  {
    \exp_args:Nc \newattribute { g__LuaCJK_#1_attr }
    \int_const:cn { c__LuaCJK_#1_attr_int } { \int_use:N \allocationnumber }
  }
\g__LuaCJK_cjk_space_attr  = 0
\g__LuaCJK_xecglue_attr    = 0
\g__LuaCJK_math_font_attr  = 0
\g__LuaCJK_punct_font_attr = 0
\g__LuaCJK_settings_attr   = 0
```

(*End of definition for* `\g__LuaCJK_font_attr` *and others.*)

\l__LuaCJK_ccglue_skip
\l__LuaCJK_ecglue_skip

The glue used between CJK characters and between CJK and Latin text.

```
\skip_new:N \l__LuaCJK_ccglue_skip
\skip_new:N \l__LuaCJK_ecglue_skip
\bool_new:N \l__LuaCJK_ccglue_touched_bool
\bool_new:N \l__LuaCJK_ecglue_touched_bool
\skip_set:Nn \l__LuaCJK_ccglue_skip { 0pt plus 0.08 \tex_baselineskip:D }
\skip_set:Nn \l__LuaCJK_ecglue_skip
  { 0.3333em plus 0.1667em minus 0.1111em }
```

(*End of definition for* `\l__LuaCJK_ccglue_skip` *and* `\l__LuaCJK_ecglue_skip`.)

__LuaCJK_set_ccglue:n
__LuaCJK_set_ecglue:n

The two glues keep their default relative to the current size, so they are recomputed on every font selection unless the document set them. A value given by the document is taken as it is.

```
\cs_new_protected:Npn \__LuaCJK_set_ccglue:n #1
  {
```

```
    \skip_set:Nn \l__LuaCJK_ccglue_skip {#1}
    \bool_set_true:N \l__LuaCJK_ccglue_touched_bool
  }
\cs_new_protected:Npn \__LuaCJK_set_ecglue:n #1
  {
    \skip_set:Nn \l__LuaCJK_ecglue_skip {#1}
    \bool_set_true:N \l__LuaCJK_ecglue_touched_bool
  }
\cs_new_protected:Npn \__LuaCJK_refresh_glues:
  {
    \bool_if:NF \l__LuaCJK_ccglue_touched_bool
      {
        \skip_set:Nn \l__LuaCJK_ccglue_skip
          { 0pt plus 0.08 \tex_baselineskip:D }
      }
    \bool_if:NF \l__LuaCJK_ecglue_touched_bool
      {
        \skip_set:Nn \l__LuaCJK_ecglue_skip
          { 0.3333em plus 0.1667em minus 0.1111em }
      }
  }
```

(*End of definition for* `\__LuaCJK_set_ccglue:n` *and* `\__LuaCJK_set_ecglue:n`.)

`\__LuaCJK_explicit_space:`
`\__LuaCJK_explicit_nobreak_space:`
`\__LuaCJK_nobreak_space:`

The space the document asked for is marked. The node level cannot tell the two apart: a source space and `\ ` reach the spacing pass as the same `spaceskip` glue with the same dimensions, so the mark is the only thing that separates them. `\ ` and `\nobreakspace` become a group around the space token, and `~` reaches the second through `\nobreakspace`. Both are protected, so an `\edef` or an `\write` carries them as one token: the body of an `\edef` is still the space the document wrote, at the price of a message printed with `\ ` showing the control space instead of a space. `\space` stays a source space, because it is the form that appears in messages and has to expand. The definition of `\nobreakspace` is copied before it is replaced, because a body that named `\nobreakspace` again would call the wrapper once the wrapper is in its place. `\scan_stop` ends the value of the attribute: a space token that came straight after it would be taken for the blank that ends the number. The group that sets the catcode of the space is what lets the two be assigned at all: expl3 ignores a space, so a control space cannot be written in its syntax.

```
\cs_new_eq:NN \__LuaCJK_nobreak_space: \nobreakspace
\cs_new_protected:Npn \__LuaCJK_explicit_space:
  {
    \group_begin:
```

```
      \g__LuaCJK_explicit_space_attr = 1 \scan_stop:
      \char_generate:nn { 32 } { 10 }
    \group_end:
  }
\cs_new_protected:Npn \__LuaCJK_explicit_nobreak_space:
  {
    \group_begin:
      \g__LuaCJK_explicit_space_attr = 1 \scan_stop:
      \__LuaCJK_nobreak_space:
    \group_end:
  }
\group_begin:
  \char_set_catcode_space:n { 32 }
  \cs_gset_eq:NN \ \__LuaCJK_explicit_space:
  \cs_gset_eq:NN \nobreakspace \__LuaCJK_explicit_nobreak_space:
\group_end:
```

(*End of definition for* `\__LuaCJK_explicit_space:`, `\__LuaCJK_explicit_nobreak_space:`, *and* `\__LuaCJK_nobreak_space:`.)

A punctuation style is a property list, read through the settings of each node.

`g__LuaCJK_punct_style_prop` The names of the styles LuaCJK knows about, including those defined by `\LuaCJKDeclarePunctStyle`.

```
\prop_new:N \g__LuaCJK_punct_style_prop
\prop_gput:Nnn \g__LuaCJK_punct_style_prop { quanjiao      } { }
\prop_gput:Nnn \g__LuaCJK_punct_style_prop { plain         } { }
\prop_gput:Nnn \g__LuaCJK_punct_style_prop { banjiao       } { }
\prop_gput:Nnn \g__LuaCJK_punct_style_prop { kaiming       } { }
\prop_gput:Nnn \g__LuaCJK_punct_style_prop { CCT           } { }
\prop_gput:Nnn \g__LuaCJK_punct_style_prop { hangmobanjiao } { }
```

(*End of definition for* `\g__LuaCJK_punct_style_prop`.)

`__LuaCJK_set_punct_style:n` Store the selected style, or warn and fall back to `plain` when the name is unknown. The style name read by the Lua side.

`\l__LuaCJK_punct_style_tl`

```
\tl_new:N \l__LuaCJK_punct_style_tl
\tl_set:Nn \l__LuaCJK_punct_style_tl { quanjiao }

\msg_new:nnn { LuaCJK } { punct-style-unknown }
  {
    PunctStyle ~ `#1' ~ is ~ not ~ implemented.\\
    Using ~ `plain' ~ instead.
  }
\cs_new_protected:Npn \__LuaCJK_set_punct_style:n #1
  {
```

```
    \prop_if_in:NnTF \g__LuaCJK_punct_style_prop {#1}
      { \tl_set:Nn \l__LuaCJK_punct_style_tl {#1} }
      {
        \msg_warning:nnn { LuaCJK } { punct-style-unknown } {#1}
        \tl_set:Nn \l__LuaCJK_punct_style_tl { plain }
      }
  }
```

(*End of definition for* `\__LuaCJK_set_punct_style:n` *and* `\l__LuaCJK_punct_style_tl`.)

`\LuaCJKEditPunctStyle` Change the parameters of an existing style. A key that is not given keeps its current value, which is why the three "given" switches are tracked here instead of starting from the defaults. Whether the corresponding key was part of the edit.

```
\bool_new:N \l__LuaCJK_punct_compress_given_bool
\bool_new:N \l__LuaCJK_punct_ratio_given_bool
\bool_new:N \l__LuaCJK_punct_keep_given_bool
\tl_new:N \l__LuaCJK_punct_compress_value_tl
\tl_new:N \l__LuaCJK_punct_keep_value_tl

\keys_define:nn { LuaCJK / punct-edit }
  {
    compress-adjacent .code:n =
      {
        \bool_set_true:N \l__LuaCJK_punct_compress_given_bool
        \tl_set:Nn \l__LuaCJK_punct_compress_value_tl {#1}
      },
    adjacent-ratio    .code:n =
      {
        \bool_set_true:N \l__LuaCJK_punct_ratio_given_bool
        \fp_set:Nn \l__LuaCJK_punct_ratio_fp {#1}
      },
    keep-open-close   .code:n =
      {
        \bool_set_true:N \l__LuaCJK_punct_keep_given_bool
        \tl_set:Nn \l__LuaCJK_punct_keep_value_tl {#1}
      },
  }
\NewDocumentCommand \LuaCJKEditPunctStyle { m m }
  {
    \bool_set_false:N \l__LuaCJK_punct_compress_given_bool
    \bool_set_false:N \l__LuaCJK_punct_ratio_given_bool
    \bool_set_false:N \l__LuaCJK_punct_keep_given_bool
    \keys_set:nn { LuaCJK / punct-edit } {#2}
    \lua_now:e
```

```
    {
      LuaCJK.edit_punct_style
        ( "\lua_escape:e {#1}" ,
          \bool_if:NTF \l__LuaCJK_punct_compress_given_bool
            { \l__LuaCJK_punct_compress_value_tl  } { nil },
          \bool_if:NTF \l__LuaCJK_punct_ratio_given_bool
            { \fp_use:N \l__LuaCJK_punct_ratio_fp } { nil },
          \bool_if:NTF \l__LuaCJK_punct_keep_given_bool
            { \l__LuaCJK_punct_keep_value_tl      } { nil } )
    }
}
```

(*End of definition for* `\LuaCJKEditPunctStyle` *and others. This function is documented on page 35.*)

`\LuaCJKsetwidth` Give specific punctuation marks a fixed advance, overriding the selected style. The star form of xeCJK, which sets the width used at the start or end of a line, has no counterpart yet and only warns. Guards the one-time warning for the star form. `\l__LuaCJK_bound_width_warned_bool`

```
\bool_new:N \l__LuaCJK_bound_width_warned_bool
\msg_new:nnn { LuaCJK } { bound-width-unsupported }
  {
    The ~ star ~ form ~ of ~ \token_to_str:N \LuaCJKsetwidth, ~ which ~
    sets ~ the ~ width ~ at ~ the ~ start ~ or ~ end ~ of ~ a ~ line, ~ is ~
    not ~ implemented.\\
    The ~ width ~ is ~ applied ~ to ~ every ~ position ~ instead.
  }

\NewDocumentCommand \LuaCJKsetwidth { s m m }
  {
    \IfBooleanT {#1}
      {
        \bool_if:NF \l__LuaCJK_bound_width_warned_bool
          {
            \bool_set_true:N \l__LuaCJK_bound_width_warned_bool
            \msg_warning:nn { LuaCJK } { bound-width-unsupported }
          }
      }
    \lua_now:e
      {
        LuaCJK.set_punct_width
          ( "\lua_escape:e {#2}" , \dim_to_decimal_in_sp:n {#3} )
      }
  }
```

(*End of definition for* `\LuaCJKsetwidth` *and* `\l__LuaCJK_bound_width_warned_bool`*. This function is documented on page 35.*)

\LuaCJKsetkern Set the gap between two adjacent punctuation marks, overriding the selected style. Only the first character of each argument is used.

```
\NewDocumentCommand \LuaCJKsetkern { m m m }
  {
    \lua_now:e
      {
        LuaCJK.set_punct_kern
          ( "\lua_escape:e {#1}" , "\lua_escape:e {#2}"
          , \dim_to_decimal_in_sp:n {#3} )
      }
  }
```

(*End of definition for* `\LuaCJKsetkern`*. This function is documented on page* 35.)

A single CJK font rarely covers every block of Unicode. A sub-block gives a range of characters its own font inside a family, named by an option of the font commands.

\LuaCJKDeclareSubCJKBlock \g__LuaCJK_block_prop \g__LuaCJK_block_names_clist The ranges of each block, and the declaration order that decides which block wins when the ranges overlap.

```
\prop_new:N \g__LuaCJK_block_prop
\clist_new:N \g__LuaCJK_block_names_clist
\NewDocumentCommand \LuaCJKDeclareSubCJKBlock { m m }
  {
    \__LuaCJK_build_ranges:n {#2}
    \prop_gput:NnV \g__LuaCJK_block_prop {#1} \l__LuaCJK_ranges_tl
    \clist_if_in:NnF \g__LuaCJK_block_names_clist {#1}
      { \clist_gput_right:Nn \g__LuaCJK_block_names_clist {#1} }
    \lua_now:e
      {
        LuaCJK.declare_block
          ( "\lua_escape:e {#1}" , { \l__LuaCJK_ranges_tl } )
      }
  }
```

(*End of definition for* `\LuaCJKDeclareSubCJKBlock`*,* `\g__LuaCJK_block_prop`*, and* `\g__LuaCJK_block_names_clist`*. This function is documented on page* 26.)

\LuaCJKCancelSubCJKBlock \LuaCJKRestoreSubCJKBlock Turn a block off or on again. The list of canceled blocks is part of the settings of the text, so the change ends with the group; the fonts of the current family are refreshed so it takes effect at once.

```
\clist_new:N \l__LuaCJK_blocks_off_clist
\cs_new_protected:Npn \__LuaCJK_set_block_enabled:nn #1#2
  {
```

```
    \clist_map_inline:nn {#1}
      {
        \str_if_eq:nnTF {#2} { false }
          {
            \clist_if_in:NnF \l__LuaCJK_blocks_off_clist {##1}
              { \clist_put_right:Nn \l__LuaCJK_blocks_off_clist {##1} }
          }
          { \clist_remove_all:Nn \l__LuaCJK_blocks_off_clist {##1} }
      }
    \__LuaCJK_settings_push:
    \__LuaCJK_update_font:
  }
\NewDocumentCommand \LuaCJKCancelSubCJKBlock { m }
  { \__LuaCJK_set_block_enabled:nn {#1} { false } }
\NewDocumentCommand \LuaCJKRestoreSubCJKBlock { m }
  { \__LuaCJK_set_block_enabled:nn {#1} { true } }
```

(*End of definition for* `\LuaCJKCancelSubCJKBlock` *and* `\LuaCJKRestoreSubCJKBlock`*. These functions are documented on page* 27.)

`\__LuaCJK_extract_font_keys:nn`
`\__LuaCJK_parse_font_options:n`

Pull the keys that LuaCJK keeps for itself out of a font option list: the block names, which name a font of their own, and `Baseline`. The remaining options go into `\l__LuaCJK_features_tl`, which is what fontspec is given, and `Baseline` goes into `\l__LuaCJK_shift_tl`. A list that names neither is handed to fontspec exactly as it was written, because the parse would only respell it. The block font of each family and block, the blocks found in the option list being processed, the filtered options, the vertical offset `Baseline` asks for, and scratch variables for splitting one option at the equals sign.

```
\prop_new:N \g__LuaCJK_family_block_prop
\prop_new:N \g__LuaCJK_pending_block_prop
\tl_new:N \l__LuaCJK_features_tl
\tl_new:N \l__LuaCJK_block_key_tl
\tl_new:N \l__LuaCJK_option_key_tl
\seq_new:N \l__LuaCJK_option_seq
\bool_new:N \l__LuaCJK_is_block_bool
\tl_new:N \l__LuaCJK_shift_tl
\cs_new_protected:Npn \__LuaCJK_put_feature:n #1
  {
    \tl_if_empty:NTF \l__LuaCJK_features_tl
      { \tl_set:Nn \l__LuaCJK_features_tl {#1} }
      { \tl_put_right:Nn \l__LuaCJK_features_tl { , ~ #1 } }
  }
\cs_new_protected:Npn \__LuaCJK_font_option:nn #1#2
  {
```

```
    \tl_set:Nn \l__LuaCJK_option_key_tl {#1}
    \tl_trim_spaces:N \l__LuaCJK_option_key_tl
    \str_if_eq:eeTF { \l__LuaCJK_option_key_tl } { Baseline }
      { \tl_set:Ne \l__LuaCJK_shift_tl {#2} }
      {
        \bool_set_false:N \l__LuaCJK_is_block_bool
        \clist_map_inline:Nn \g__LuaCJK_block_names_clist
          {
            \str_if_eq:eeT {##1} { \l__LuaCJK_option_key_tl }
              { \bool_set_true:N \l__LuaCJK_is_block_bool }
          }
        \bool_if:NTF \l__LuaCJK_is_block_bool
          { \prop_gput:NVn \g__LuaCJK_pending_block_prop
              \l__LuaCJK_option_key_tl {#2} }
          { \__LuaCJK_put_feature:n { #1 = {#2} } }
      }
  }
\cs_new_protected:Npn \__LuaCJK_font_option:n #1
  { \__LuaCJK_put_feature:n {#1} }
\cs_new_protected:Npn \__LuaCJK_parse_font_options:n #1
  {
    \tl_clear:N \l__LuaCJK_features_tl
    \tl_clear:N \l__LuaCJK_shift_tl
    \keyval_parse:NNn \__LuaCJK_font_option:n \__LuaCJK_font_option:nn {#1}
  }
\cs_new_protected:Npn \__LuaCJK_extract_font_keys:nn #1#2
  {
    \prop_gclear:N \g__LuaCJK_pending_block_prop
    \__LuaCJK_parse_font_options:n {#2}
    \prop_gput:NnV \g__LuaCJK_shift_decl_prop {#1} \l__LuaCJK_shift_tl
    \prop_if_empty:NTF \g__LuaCJK_pending_block_prop
      {
        \tl_if_empty:NT \l__LuaCJK_shift_tl
          { \tl_set:Nn \l__LuaCJK_features_tl {#2} }
      }
      { \__LuaCJK_register_block_fonts:n {#1} }
  }
\cs_new_protected:Npn \__LuaCJK_register_block_fonts:n #1
  {
    \prop_map_inline:Nn \g__LuaCJK_pending_block_prop
      {
        \group_begin:
          \__LuaCJK_build_features:n { }
          \fontspec_gset_family:NVn \l__LuaCJK_tmpa_tl
            \l__LuaCJK_feature_tl {##2}
```

```
          \tl_set:Ne \l__LuaCJK_block_key_tl { #1 | ##1 }
          \prop_gput:NVV \g__LuaCJK_family_block_prop
            \l__LuaCJK_block_key_tl \l__LuaCJK_tmpa_tl
        \group_end:
      }
  }
```

(*End of definition for* `\__LuaCJK_extract_font_keys:nn` *and others.*)

`\LuaCJKDeclareCharClass` Override the class of a code-point range. The range list uses the xeCJK spelling: hexadecimal numbers with a leading `"`, and `"START -> "END` for a range. The Lua table being built, the two halves of one range, and the current hexadecimal number and its value.

```
\tl_new:N \l__LuaCJK_ranges_tl
\seq_new:N \l__LuaCJK_range_seq
\tl_new:N \l__LuaCJK_range_pair_tl
\tl_new:N \l__LuaCJK_hex_tl
\tl_new:N \l__LuaCJK_hex_int_tl

\cs_new_protected:Npn \__LuaCJK_hex_store:n #1
  {
    \tl_set:Nn \l__LuaCJK_hex_tl {#1}
    \tl_replace_all:Nnn \l__LuaCJK_hex_tl { " } { }
    \tl_trim_spaces:N \l__LuaCJK_hex_tl
    \tl_set:Ne \l__LuaCJK_hex_int_tl
      { \exp_args:NV \int_from_hex:n \l__LuaCJK_hex_tl }
  }
\cs_new_protected:Npn \__LuaCJK_range_entry:n #1
  {
    \seq_set_split:Nnn \l__LuaCJK_range_seq { -> } {#1}
    \tl_clear:N \l__LuaCJK_range_pair_tl
    \seq_map_inline:Nn \l__LuaCJK_range_seq
      {
        \__LuaCJK_hex_store:n {##1}
        \tl_put_right:Nx \l__LuaCJK_range_pair_tl
          { \exp_not:V \l__LuaCJK_hex_int_tl , }
      }
    \tl_put_right:Nx \l__LuaCJK_ranges_tl
      { { \exp_not:V \l__LuaCJK_range_pair_tl } , }
  }
\cs_new_protected:Npn \__LuaCJK_build_ranges:n #1
  {
    \tl_clear:N \l__LuaCJK_ranges_tl
    \clist_map_inline:nn {#1} { \__LuaCJK_range_entry:n {##1} }
  }
```

```
\msg_new:nnn { LuaCJK } { unknown-char-class }
  { Unknown ~ character   ~ class ~ `#1' ~ is ~ being ~ ignored. }
\msg_new:nnn { LuaCJK } { unknown-punct-class }
  { Unknown ~ punctuation ~ class ~ `#1' ~ is ~ being ~ ignored. }
\NewDocumentCommand \LuaCJKDeclareCharClass { m m }
  {
    \str_case:nnF {#1}
      {
        { CJK         } { \__LuaCJK_push_class:nn { true    } {#2} }
        { Default     } { \__LuaCJK_push_class:nn { false   } {#2} }
        { FullLeft    } { \__LuaCJK_push_punct:nn { "open"  } {#2} }
        { HalfLeft    } { \__LuaCJK_push_punct:nn { "half-
open" } {#2} }
        { FullRight   } { \__LuaCJK_push_punct:nn { "close" } {#2} }
        { HalfRight   } { \__LuaCJK_push_punct:nn { "half-
close" } {#2} }
        { NormalSpace } { \__LuaCJK_push_normal_space:n       {#2} }
        { CJStarter   } { \__LuaCJK_push_class:nn { true    } {#2} }
        { CM          } { \__LuaCJK_push_class:nn { true    } {#2} }
        { Boundary    } { \__LuaCJK_push_class:nn { true    } {#2} }
      } { \msg_warning:nnn { LuaCJK } { unknown-char-class } {#1} }
  }
\cs_new_protected:Npn \__LuaCJK_push_normal_space:n #1
  {
    \__LuaCJK_build_ranges:n {#1}
    \lua_now:e
      { LuaCJK.add_normal_spaced ( { \l__LuaCJK_ranges_tl } ) }
  }
\cs_new_protected:Npn \__LuaCJK_push_class:nn #1#2
  {
    \__LuaCJK_build_ranges:n {#2}
    \lua_now:e
      { LuaCJK.declare_class ( { \l__LuaCJK_ranges_tl } , #1 ) }
  }
```

(*End of definition for* `\LuaCJKDeclareCharClass` *and others. This function is documented on page* 34.)

`\LuaCJKDeclarePunctClass` Override the punctuation group of a range, or with `none` remove the punctuation treatment from it.

```
\NewDocumentCommand \LuaCJKDeclarePunctClass { m m }
  {
    \str_case:nnF {#1}
      {
        { open     } { \__LuaCJK_push_punct:nn { "open" } {#2}     }
        { close    } { \__LuaCJK_push_punct:nn { "close" } {#2}    }
```

```
        { middle   } { \__LuaCJK_push_punct:nn { "middle" } {#2}   }
        { dash     } { \__LuaCJK_push_punct:nn { "dash" } {#2}     }
        { ellipsis } { \__LuaCJK_push_punct:nn { "ellipsis" } {#2} }
        { half-open  } { \__LuaCJK_push_punct:nn { "half-
open" } {#2} }
        { half-close } { \__LuaCJK_push_punct:nn { "half-
close" } {#2} }
        { none     } { \__LuaCJK_push_punct:nn { false } {#2}      }
      } { \msg_warning:nnn { LuaCJK } { unknown-punct-class } {#1} }
  }
\cs_new_protected:Npn \__LuaCJK_push_punct:nn #1#2
  {
    \__LuaCJK_build_ranges:n {#2}
    \lua_now:e
      { LuaCJK.declare_punct ( { \l__LuaCJK_ranges_tl } , #1 ) }
  }
```

(*End of definition for* `\LuaCJKDeclarePunctClass`*. This function is documented on page 35.*)

`\__LuaCJK_set_charset:nnn` Replace, extend, or reduce one of the punctuation character sets. The ranges are converted the same way as for `\LuaCJKDeclareCharClass`.

```
\cs_new_protected:Npn \__LuaCJK_set_charset:nnn #1#2#3
  {
    \__LuaCJK_build_ranges:n {#3}
    \lua_now:e
      {
        LuaCJK.set_charset
          ( "#1" , "#2" , { \l__LuaCJK_ranges_tl } )
      }
  }
```

(*End of definition for* `\__LuaCJK_set_charset:nnn`.)

`\LuaCJKResetCharClass` Drop every class and punctuation override declared so far, and put the normal-spaced set back to the characters of `\normalspacedchars`.

```
\NewDocumentCommand \LuaCJKResetCharClass { }
  {
    \lua_now:n { LuaCJK.reset_classes() }
    \__LuaCJK_set_normal_spaced_chars:
  }
```

(*End of definition for* `\LuaCJKResetCharClass`*. This function is documented on page 35.*)

`\LuaCJKDeclarePunctStyle` Define a style from a small set of keys and hand the result to Lua. The values collected from the keys, with the defaults used when a key is not given.

```
\bool_new:N \l__LuaCJK_punct_compress_bool
```

```
\fp_new:N \l__LuaCJK_punct_ratio_fp
\bool_new:N \l__LuaCJK_punct_keep_bool

\keys_define:nn { LuaCJK / punct-style }
  {
    compress-adjacent .bool_set:N = \l__LuaCJK_punct_compress_bool,
    adjacent-ratio    .fp_set:N   = \l__LuaCJK_punct_ratio_fp,
    keep-open-close   .bool_set:N = \l__LuaCJK_punct_keep_bool,
  }
\NewDocumentCommand \LuaCJKDeclarePunctStyle { m m }
  {
    \bool_set_true:N \l__LuaCJK_punct_compress_bool
    \fp_set:Nn \l__LuaCJK_punct_ratio_fp { 0.5 }
    \bool_set_true:N \l__LuaCJK_punct_keep_bool
    \keys_set:nn { LuaCJK / punct-style } {#2}
    \prop_gput:Nnn \g__LuaCJK_punct_style_prop {#1} { }
    \lua_now:e
      {
        LuaCJK.define_punct_style(
          "\lua_escape:e {#1}",
          \bool_if:NTF \l__LuaCJK_punct_compress_bool { true } { false },
          \fp_use:N \l__LuaCJK_punct_ratio_fp,
          \bool_if:NTF \l__LuaCJK_punct_keep_bool { true } { false } )
      }
  }
```

(*End of definition for* `\LuaCJKDeclarePunctStyle` *and others. This function is documented on page 35.*)

`\__LuaCJK_set_autofake:nn` Turn fake bold or slant on or off. A number both enables the feature and sets the factor, as in xeCJK.

```
\cs_new_protected:Npn \__LuaCJK_set_autofake:nn #1#2
  {
    \str_case:nnF {#2}
      {
        { true  }
        {
          \str_if_eq:nnTF {#1} { bold }
            {
              \tl_gset_eq:NN \g__LuaCJK_autofake_bold_tl
                \g__LuaCJK_embolden_factor_tl
            }
            {
              \tl_gset_eq:NN \g__LuaCJK_autofake_slant_tl
                \g__LuaCJK_slant_factor_tl
            }
```

```
        }
        { false }
        {
          \str_if_eq:nnTF {#1} { bold }
            { \tl_gclear:N \g__LuaCJK_autofake_bold_tl  }
            { \tl_gclear:N \g__LuaCJK_autofake_slant_tl }
        }
      }
      {
        \str_if_eq:nnTF {#1} { bold }
          {
            \tl_gset:Nn \g__LuaCJK_embolden_factor_tl {#2}
            \tl_gset:Nn \g__LuaCJK_autofake_bold_tl   {#2}
          }
          {
            \tl_gset:Nn \g__LuaCJK_slant_factor_tl    {#2}
            \tl_gset:Nn \g__LuaCJK_autofake_slant_tl  {#2}
          }
      }
  }
```

(*End of definition for* `\__LuaCJK_set_autofake:nn`.)

`\LuaCJKsetup` The same key path serves `\LuaCJKsetup` and the package options. The keys are defined here, but the package options are read at the end of the package, once every helper a key code calls has been defined and the Lua module has been loaded, which `CJLineBreak` needs: a key whose code calls a helper of the font section would otherwise act through `\LuaCJKsetup` alone and leave an undefined control sequence behind when it was given as a package option. The switches read by the Lua side, the `Verb` setting, whether a block of verbatim material is being set, the configuration file name, the fake bold and slant parameters, and the fallback fonts.

```
\tl_new:N  \l__LuaCJK_punct_kinsoku_tl
\tl_set:Nn \l__LuaCJK_punct_kinsoku_tl { 1 }
\tl_new:N  \l__LuaCJK_cj_strict_tl
\tl_set:Nn \l__LuaCJK_cj_strict_tl { 0 }
\tl_new:N  \l__LuaCJK_cjk_math_tl
\tl_set:Nn \l__LuaCJK_cjk_math_tl { 0 }
\tl_new:N  \l__LuaCJK_latin_punct_tl
\tl_set:Nn \l__LuaCJK_latin_punct_tl { 0 }
\tl_new:N  \l__LuaCJK_verb_mode_tl
\tl_set:Nn \l__LuaCJK_verb_mode_tl { true }
\bool_new:N \l__LuaCJK_verbatim_block_bool
\bool_new:N \l__LuaCJK_break_between_puncts_bool
```

```
\dim_new:N  \l__LuaCJK_punct_width_dim
\dim_set:Nn \l__LuaCJK_punct_width_dim { 0pt }
\bool_new:N \l__LuaCJK_punct_width_given_bool
\dim_new:N  \l__LuaCJK_punct_bound_width_dim
\dim_set:Nn \l__LuaCJK_punct_bound_width_dim { 0pt }
\bool_new:N \l__LuaCJK_punct_bound_width_given_bool
\tl_new:N \l__LuaCJK_check_single_tl
\tl_set:Nn \l__LuaCJK_check_single_tl { 0 }
\int_new:N \l__LuaCJK_widow_penalty_int
\int_set:Nn \l__LuaCJK_widow_penalty_int { 10000 }
\tl_new:N  \l__LuaCJK_local_config_tl
\tl_new:N  \g__LuaCJK_autofake_bold_tl
\tl_new:N  \g__LuaCJK_autofake_slant_tl
\tl_new:N  \g__LuaCJK_embolden_factor_tl
\tl_gset:Nn \g__LuaCJK_embolden_factor_tl { 4 }
\tl_new:N  \g__LuaCJK_slant_factor_tl
\tl_gset:Nn \g__LuaCJK_slant_factor_tl { 0.167 }
\prop_new:N \g__LuaCJK_fallback_prop
\bool_new:N \l__LuaCJK_auto_fallback_bool
\bool_new:N \g__LuaCJK_load_fandol_bool
\tl_new:N \l__LuaCJK_fallback_family_tl
\tl_new:N \l__LuaCJK_fallback_key_tl
\prop_new:N \g__LuaCJK_shift_decl_prop
\prop_new:N \g__LuaCJK_font_shift_prop
\prop_new:N \g__LuaCJK_shift_pushed_prop
\tl_new:N \l__LuaCJK_shift_dim_tl
\tl_new:N \l__LuaCJK_shift_fid_tl
\tl_new:N \l__LuaCJK_shift_key_tl
\tl_new:N \l__LuaCJK_shift_value_tl
\tl_new:N \l__LuaCJK_shift_old_tl
\cs_generate_variant:Nn \prop_get:NnNF { NVNF }
\cs_generate_variant:Nn \prop_gput:Nnn { NVV , NnV }
\tl_new:N \l__LuaCJK_fallback_ids_tl
\tl_new:N \l__LuaCJK_fontid_tl

\keys_define:nn { LuaCJK / options }
  {
    CJKglue             .code:n      = { \__LuaCJK_set_ccglue:n {#1} },
    CJKecglue           .code:n      = { \__LuaCJK_set_ecglue:n {#1} },
    CJKspace            .choice:,
    CJKspace / true     .code:n      = { \g__LuaCJK_cjk_space_attr = 1 },
    CJKspace / false    .code:n      = { \g__LuaCJK_cjk_space_attr = 0 },
    CJKspace            .default:n   = { true  },
    CJKspace            .initial:n   = { false },
    xCJKecglue          .choice:,
```

```
    xCJKecglue / true   .code:n      = { \g__LuaCJK_xecglue_attr = 1 },
    xCJKecglue / false  .code:n      = { \g__LuaCJK_xecglue_attr = 0 },
    xCJKecglue          .default:n   = { true  },
    xCJKecglue          .initial:n   = { false },
    PunctStyle          .code:n      = { \__LuaCJK_set_punct_style:n {#1} },
    PunctStyle          .default:n   = { quanjiao },
    PunctKinsoku        .choice:,
    PunctKinsoku / true .code:n      =
      { \tl_set:Nn \l__LuaCJK_punct_kinsoku_tl { 1 } },
    PunctKinsoku / false.code:n      =
      { \tl_set:Nn \l__LuaCJK_punct_kinsoku_tl { 0 } },
    PunctKinsoku        .default:n   = { true  },
    CJLineBreak         .choice:,
    CJLineBreak / normal.code:n      =
      { \tl_set:Nn \l__LuaCJK_cj_strict_tl { 0 } },
    CJLineBreak / strict.code:n      =
      {
        \tl_set:Nn \l__LuaCJK_cj_strict_tl { 1 }
        \__LuaCJK_load_cj:
      },
    CJLineBreak         .default:n   = { strict },
    CJKmath             .choice:,
    CJKmath / true      .code:n      =
      { \tl_set:Nn \l__LuaCJK_cjk_math_tl { 1 } \__LuaCJK_update_font: },
    CJKmath / false     .code:n      =
      { \tl_set:Nn \l__LuaCJK_cjk_math_tl { 0 } \__LuaCJK_update_font: },
    CJKmath             .default:n   = { true  },
    Verb                .choice:,
    Verb / true         .code:n      =
      { \tl_set:Nn \l__LuaCJK_verb_mode_tl { true  } },
    Verb / env          .code:n      =
      { \tl_set:Nn \l__LuaCJK_verb_mode_tl { env   } },
    Verb / env+         .code:n      =
      { \tl_set:Nn \l__LuaCJK_verb_mode_tl { env+  } },
    Verb / false        .code:n      =
      { \tl_set:Nn \l__LuaCJK_verb_mode_tl { false } },
    Verb                .default:n   = { true },
    KaiMingPunct        .code:n      =
      { \__LuaCJK_set_charset:nnn { kaiming } { set    } {#1} },
    KaiMingPunct+       .code:n      =
      { \__LuaCJK_set_charset:nnn { kaiming } { add    } {#1} },
    KaiMingPunct-       .code:n      =
      { \__LuaCJK_set_charset:nnn { kaiming } { remove } {#1} },
    MiddlePunct         .code:n      =
      { \__LuaCJK_set_charset:nnn { middle  } { set    } {#1} },
```

```
    MiddlePunct+        .code:n      =
      { \__LuaCJK_set_charset:nnn { middle  } { add    } {#1} },
    MiddlePunct-        .code:n      =
      { \__LuaCJK_set_charset:nnn { middle  } { remove } {#1} },
    LongPunct           .code:n      =
      { \__LuaCJK_set_charset:nnn { long    } { set    } {#1} },
    LongPunct+          .code:n      =
      { \__LuaCJK_set_charset:nnn { long    } { add    } {#1} },
    LongPunct-          .code:n      =
      { \__LuaCJK_set_charset:nnn { long    } { remove } {#1} },
    NoBreakLongPunct    .code:n      =
      { \__LuaCJK_set_charset:nnn { nobreak } { set    } {#1} },
    NoBreakLongPunct+   .code:n      =
      { \__LuaCJK_set_charset:nnn { nobreak } { add    } {#1} },
    NoBreakLongPunct-   .code:n      =
      { \__LuaCJK_set_charset:nnn { nobreak } { remove } {#1} },
    PunctWidth          .code:n      =
      {
        \str_if_eq:nnTF {#1} { false }
          {
            \bool_set_false:N \l__LuaCJK_punct_width_given_bool
            \dim_zero:N \l__LuaCJK_punct_width_dim
          }
          {
            \bool_set_true:N \l__LuaCJK_punct_width_given_bool
            \dim_set:Nn \l__LuaCJK_punct_width_dim {#1}
          }
      },
    PunctWidth          .default:n   = { false },
    PunctFamily         .code:n      =
      {
        \str_if_eq:nnTF {#1} { false }
          { \tl_clear:N \l__LuaCJK_punct_family_tl }
          { \tl_set:Nn \l__LuaCJK_punct_family_tl {#1} }
        \__LuaCJK_update_font:
      },
    PunctFamily         .default:n   = { false },
    PunctBoundWidth     .code:n      =
      {
        \str_if_eq:nnTF {#1} { false }
          {
            \bool_set_false:N \l__LuaCJK_punct_bound_width_given_bool
            \dim_zero:N \l__LuaCJK_punct_bound_width_dim
          }
          {
```

```
            \bool_set_true:N \l__LuaCJK_punct_bound_width_given_bool
            \dim_set:Nn \l__LuaCJK_punct_bound_width_dim {#1}
          }
      },
    PunctBoundWidth     .default:n   = { false },
    LatinPunct          .choice:,
    LatinPunct / true   .code:n      =
      { \tl_set:Nn \l__LuaCJK_latin_punct_tl { 1 } },
    LatinPunct / false  .code:n      =
      { \tl_set:Nn \l__LuaCJK_latin_punct_tl { 0 } },
    LatinPunct          .default:n   = { true  },
    LatinPunct          .initial:n   = { false },
    AllowBreakBetweenPuncts
                        .choice:,
    AllowBreakBetweenPuncts / true
                        .code:n      =
      { \bool_set_true:N \l__LuaCJK_break_between_puncts_bool },
    AllowBreakBetweenPuncts / false
                        .code:n      =
      { \bool_set_false:N \l__LuaCJK_break_between_puncts_bool },
    AllowBreakBetweenPuncts
                        .default:n   = { true  },
    CheckSingle         .choice:,
    CheckSingle / true  .code:n      =
      { \tl_set:Nn \l__LuaCJK_check_single_tl { 1 } },
    CheckSingle / false .code:n      =
      { \tl_set:Nn \l__LuaCJK_check_single_tl { 0 } },
    CheckSingle         .default:n   = { true },
    WidowPenalty        .code:n      =
      { \int_set:Nn \l__LuaCJK_widow_penalty_int {#1} },
    LocalConfig         .code:n      =
      {
        \str_case:nnF {#1}
          {
            { true  } { \tl_set:Nn \l__LuaCJK_local_config_tl { luacjk.cfg } }
            { false } { \tl_clear:N \l__LuaCJK_local_config_tl          }
          } { \tl_set:Nn \l__LuaCJK_local_config_tl { luacjk-
#1.cfg } }
      },
    LocalConfig         .default:n   = { true  },
    AutoFakeBold        .code:n      = { \__LuaCJK_set_autofake:nn { bold  } {#1} },
    AutoFakeBold        .default:n   = { true  },
    AutoFakeBold        .initial:n   = { false },
    AutoFakeSlant       .code:n      = { \__LuaCJK_set_autofake:nn { slant } {#1} },
    AutoFakeSlant       .default:n   = { true  },
```

```
    AutoFakeSlant       .initial:n   = { false },
    EmboldenFactor      .code:n      =
      {
        \tl_gset:Nn \g__LuaCJK_embolden_factor_tl {#1}
        \tl_if_empty:NF \g__LuaCJK_autofake_bold_tl
          { \tl_gset:Nn \g__LuaCJK_autofake_bold_tl {#1} }
      },
    AutoFallBack        .choice:,
    AutoFallBack / true .code:n      =
      {
        \bool_set_true:N \l__LuaCJK_auto_fallback_bool
        \__LuaCJK_refresh_fallbacks:
      },
    AutoFallBack / false.code:n      =
      { \bool_set_false:N \l__LuaCJK_auto_fallback_bool },
    AutoFallBack        .default:n   = { true  },
    LoadFandol          .choice:,
    LoadFandol / true   .code:n      =
      { \bool_gset_true:N \g__LuaCJK_load_fandol_bool },
    LoadFandol / false  .code:n      =
      { \bool_gset_false:N \g__LuaCJK_load_fandol_bool },
    LoadFandol          .default:n   = { true  },
    LoadFandol          .initial:n   = { true  },
    SlantFactor         .code:n      =
      {
        \tl_gset:Nn \g__LuaCJK_slant_factor_tl {#1}
        \tl_if_empty:NF \g__LuaCJK_autofake_slant_tl
          { \tl_gset:Nn \g__LuaCJK_autofake_slant_tl {#1} }
      },
    unknown             .code:n      =
      {
        \msg_warning:nne { LuaCJK } { unknown-option }
          { \l_keys_key_str }
      },
  }
\msg_new:nnn { LuaCJK } { unknown-option }
  { Unknown ~ option ~ `#1' ~ is ~ being ~ ignored. }
\NewDocumentCommand \LuaCJKsetup { m }
  {
    \keys_set:nn { LuaCJK / options } {#1}
    \__LuaCJK_settings_push:
  }
```

(*End of definition for* `\LuaCJKsetup` *and others. This function is documented on page* 23.)

__LuaCJK_settings_push: Hand the current options to Lua, which stores them once and selects the stored copy with an attribute. Called after \LuaCJKsetup and after every font selection, because the default glues and the size-dependent distances have to be recomputed for the new size. The attribute is what makes a group that ends before the paragraph still decide how its own text is set.

```
\cs_new:Npn \__LuaCJK_bool_js:N #1
  { \str_if_eq:VnTF #1 { 1 } { true } { false } }
\cs_new:Npn \__LuaCJK_width_js:Nn #1#2
  { \bool_if:NTF #1 { \dim_to_decimal_in_sp:n {#2} } { false } }
\cs_new_protected:Npn \__LuaCJK_settings_push:
  {
    \bool_if:NT \g__LuaCJK_lua_loaded_bool
      {
        \lua_now:e
          {
            LuaCJK.new_settings
              (
                "\lua_escape:e { \l__LuaCJK_punct_style_tl }" ,
                \__LuaCJK_bool_js:N \l__LuaCJK_punct_kinsoku_tl ,
                \__LuaCJK_bool_js:N \l__LuaCJK_cj_strict_tl ,
                \__LuaCJK_bool_js:N \l__LuaCJK_latin_punct_tl ,
                \__LuaCJK_bool_js:N \l__LuaCJK_cjk_math_tl ,
                \__LuaCJK_width_js:Nn
                  \l__LuaCJK_punct_width_given_bool \l__LuaCJK_punct_width_dim ,
                \__LuaCJK_width_js:Nn
                  \l__LuaCJK_punct_bound_width_given_bool
                  \l__LuaCJK_punct_bound_width_dim ,
                \__LuaCJK_bool_js:N \l__LuaCJK_check_single_tl ,
                \int_use:N \l__LuaCJK_widow_penalty_int ,
                \bool_if:NTF \l__LuaCJK_break_between_puncts_bool
                  { true } { false } ,
                \bool_if:NTF \l__LuaCJK_auto_fallback_bool
                  { true } { false } ,
                "\lua_escape:e { \l__LuaCJK_blocks_off_clist }"
              )
          }
      }
  }
```

(*End of definition for* __LuaCJK_settings_push:.)

\LuaCJKdisable \LuaCJKenable Set the node attribute that makes the Lua callbacks skip the adjustments. The attribute follows the TEX grouping.

```
\NewDocumentCommand \LuaCJKdisable { }
```

```
  { \g__LuaCJK_disabled_attr = 1 }
\NewDocumentCommand \LuaCJKenable { }
  { \g__LuaCJK_disabled_attr = 0 }
```

(*End of definition for* `\LuaCJKdisable` *and* `\LuaCJKenable`*. These functions are documented on page 37.*)

`\LuaCJKShipoutHook` Run code just before every page is shipped out. This is the place for work that must see the finished page; xeCJK calls a macro at the same moment, so the name is kept, but the code goes into the `shipout/before` hook and therefore follows its rules.

```
\NewDocumentCommand \LuaCJKShipoutHook { m }
  { \hook_gput_code:nnn { shipout / before } { LuaCJK } {#1} }
```

(*End of definition for* `\LuaCJKShipoutHook`*. This function is documented on page 37.*)

`\LuaCJKnobreak` Keep the argument on one line. The nodes of the argument are marked, and the line-breaking pass forbids every break opportunity that involves a marked node, whether it is a glue, a kern, or a discretionary. The glue need not be one LuaCJK inserted: a glue that carries the mark forbids its break, so `\quad`, `\hspace`, and `\hfill` keep their place on the line.

```
\NewDocumentCommand \LuaCJKnobreak { +m }
  {
    \group_begin:
      \g__LuaCJK_nobreak_attr = 1
      #1
    \group_end:
  }
```

(*End of definition for* `\LuaCJKnobreak`*. This function is documented on page 37.*)

`\LuaCJKVerbAddon` `\LuaCJKOffVerbAddon` Align CJK text in a monospaced context: one CJK character plus the glue that follows it should take the width of two spaces. The gap therefore is twice the space of the current font minus the width of one CJK character. The width of the character is measured after switching to the CJK family in a group, because the current font is the monospaced one. A proportional font has no fixed space to align to, so the command adds nothing there. The glue is inserted by the spacing pass, and that pass leaves a suspended node alone, so the addon turns the adjustments back on inside its group. Without that it computes a gap that nothing inserts, which is what used to happen in a verbatim-like environment, whose begin hook suspends the material before the `formatcom` option of fancyvrb runs. The suspension the addon found is remembered, and `\LuaCJKOffVerbAddon` puts it back, so the material is left alone again once the addon is cancelled. The width



```
\l__LuaCJK_verb_space_dim
\l__LuaCJK_verb_cjk_dim
\l__LuaCJK_verb_gap_dim
\l__LuaCJK_verb_saved_ccglue_skip
\l__LuaCJK_verb_saved_ecglue_skip
\l__LuaCJK_verb_saved_ccglue_touched_bool
\l__LuaCJK_verb_saved_ecglue_touched_bool
\l__LuaCJK_verb_addon_bool
```

of two spaces, the width of one CJK character, the gap between them, the saved glues, whether the addon is on, the suspension in force when it was switched on, and the box used to measure the character.

```
\dim_new:N \l__LuaCJK_verb_space_dim
\dim_new:N \l__LuaCJK_verb_cjk_dim
\dim_new:N \l__LuaCJK_verb_gap_dim
\skip_new:N \l__LuaCJK_verb_saved_ccglue_skip
\skip_new:N \l__LuaCJK_verb_saved_ecglue_skip
\bool_new:N \l__LuaCJK_verb_saved_ccglue_touched_bool
\bool_new:N \l__LuaCJK_verb_saved_ecglue_touched_bool
\bool_new:N \l__LuaCJK_verb_addon_bool
\int_new:N \l__LuaCJK_verb_saved_disabled_int
\box_new:N \l__LuaCJK_verb_box

\msg_new:nnn { LuaCJK } { verb-addon-narrow }
  {
    The ~ monospaced ~ font ~ is ~ too ~ narrow ~ for ~ one ~ CJK ~
    character.\\
    `\token_to_str:N \LuaCJKVerbAddon' ~ adds ~ no ~ space.
  }
\cs_new_protected:Npn \__LuaCJK_verb_addon_compute:
  {
    \dim_set:Nn \l__LuaCJK_verb_space_dim
      { 2 \tex_fontdimen:D 2 \tex_font:D }
    \dim_compare:nNnTF
      { \tex_fontdimen:D 2 \tex_font:D }
      = { \tex_fontcharwd:D \tex_font:D 77 }
      {
        \dim_set_eq:NN \l__LuaCJK_verb_cjk_dim \l__LuaCJK_verb_space_dim
        \group_begin:
          \prop_get:NVNT \g__LuaCJK_family_prop \l__LuaCJK_family_tl \l__LuaCJK_tmpa_t
            {
              \exp_args:NV \__LuaCJK_family_font:n \l__LuaCJK_tmpa_tl
              \selectfont
              \hbox_set:Nn \l__LuaCJK_verb_box { 一 }
              \dim_gset:Nn \l__LuaCJK_verb_cjk_dim
                { \box_wd:N \l__LuaCJK_verb_box }
            }
        \group_end:
        \dim_set:Nn \l__LuaCJK_verb_gap_dim
          { \l__LuaCJK_verb_space_dim - \l__LuaCJK_verb_cjk_dim }
        \dim_compare:nNnTF { \l__LuaCJK_verb_gap_dim } < { 0pt }
          {
            \msg_warning:nn { LuaCJK } { verb-addon-narrow }
            \LuaCJKsetup { CJKglue = 0pt, CJKecglue = 0pt }
```

```
          }
          {
            \LuaCJKsetup
              {
                CJKglue   = \dim_use:N \l__LuaCJK_verb_gap_dim ,
                CJKecglue = \dim_eval:n { \l__LuaCJK_verb_gap_dim / 2 }
              }
          }
      }
      { \LuaCJKsetup { CJKglue = 0pt, CJKecglue = 0pt } }
  }
\NewDocumentCommand \LuaCJKVerbAddon { }
  {
    \int_compare:nNnF \tex_currentgrouplevel:D = \c_zero_int
      {
        \skip_set_eq:NN \l__LuaCJK_verb_saved_ccglue_skip
          \l__LuaCJK_ccglue_skip
        \skip_set_eq:NN \l__LuaCJK_verb_saved_ecglue_skip
          \l__LuaCJK_ecglue_skip
        \bool_set_eq:NN \l__LuaCJK_verb_saved_ccglue_touched_bool
          \l__LuaCJK_ccglue_touched_bool
        \bool_set_eq:NN \l__LuaCJK_verb_saved_ecglue_touched_bool
          \l__LuaCJK_ecglue_touched_bool
        \int_set:Nn \l__LuaCJK_verb_saved_disabled_int
          { \g__LuaCJK_disabled_attr }
        \LuaCJKenable
        \__LuaCJK_verb_addon_compute:
        \g__LuaCJK_nobreak_attr = 1
        \bool_set_true:N \l__LuaCJK_verb_addon_bool
      }
  }
\NewDocumentCommand \LuaCJKOffVerbAddon { }
  {
    \bool_if:NT \l__LuaCJK_verb_addon_bool
      {
        \skip_set_eq:NN \l__LuaCJK_ccglue_skip
          \l__LuaCJK_verb_saved_ccglue_skip
        \skip_set_eq:NN \l__LuaCJK_ecglue_skip
          \l__LuaCJK_verb_saved_ecglue_skip
        \bool_set_eq:NN \l__LuaCJK_ccglue_touched_bool
          \l__LuaCJK_verb_saved_ccglue_touched_bool
        \bool_set_eq:NN \l__LuaCJK_ecglue_touched_bool
          \l__LuaCJK_verb_saved_ecglue_touched_bool
        \__LuaCJK_settings_push:
        \g__LuaCJK_nobreak_attr = 0
```

```
      \int_compare:nNnTF { \l__LuaCJK_verb_saved_disabled_int } > { 0 }
        { \LuaCJKdisable }
        { \LuaCJKenable }
      \bool_set_false:N \l__LuaCJK_verb_addon_bool
    }
}
```

(*End of definition for* `\LuaCJKVerbAddon` *and others. These functions are documented on page 38.*)

`\__LuaCJK_verbatim@font_hook:`

Verbatim-like material must usually be left alone. `\verb` opens its own group and selects `\verbatim@font` inside it, so extending that command covers `\verb` on its own. The hook consults the current `Verb` setting, so that `\LuaCJKsetup` can change the behavior after the package is loaded.

```
\cs_new_protected:Npn \__LuaCJK_verbatim@font_hook:
  {
    \tl_if_eq:VnTF \l__LuaCJK_verb_mode_tl { true }
      { \LuaCJKdisable }
      {
        \tl_if_eq:VnT \l__LuaCJK_verb_mode_tl { env }
          { \LuaCJKdisable }
      }
  }
\cs_if_exist:NT \verbatim@font
  {
    \cs_gset_protected:Npx \verbatim@font
      {
        \exp_not:o { \verbatim@font }
        \exp_not:N \__LuaCJK_verbatim@font_hook:
      }
  }
```

(*End of definition for* `\__LuaCJK_verbatim@font_hook:`.)

`\__LuaCJK_verbatim_block_disable:`
`\__LuaCJK_fv_prep_hook:`
`\__LuaCJK_fv_patch:`

fancyvrb never selects `\verbatim@font`, and its formatting hook serves the inline commands and every block at once, so the hook waits for the package to be loaded and then applies the rule of the call it serves: the rule of `\verb` while no block is open, and the rule of the environments, which only `true` suspends, when one is. A block is an environment of the list below, or a command that sets saved verbatim material, and both mark themselves in their own group. The inline commands cannot mark themselves: a token placed after `\Verb` is scanned as part of its text, because the scanner reads to the end of the line, and `\FVC@Verb` closes its group from inside the box it sets, so neither can be wrapped. The bool is therefore false by default and set by the block side.

```
\cs_new_protected:Npn \__LuaCJK_verbatim_block_disable:
```

```
  {
    \tl_if_eq:VnT \l__LuaCJK_verb_mode_tl { true }
      { \LuaCJKdisable }
  }
\cs_new_protected:Npn \__LuaCJK_fv_prep_hook:
  {
    \bool_if:NTF \l__LuaCJK_verbatim_block_bool
      { \__LuaCJK_verbatim_block_disable: }
      { \__LuaCJK_verbatim@font_hook: }
  }
\cs_new_protected:Npn \__LuaCJK_fv_patch:
  {
    \cs_if_exist:NT \FV@FormattingPrep
      {
        \cs_gset_protected:Npx \FV@FormattingPrep
          {
            \exp_not:N \__LuaCJK_fv_prep_hook:
            \exp_not:o { \FV@FormattingPrep }
          }
      }
    \cs_if_exist:NT \FV@UseVerbatim
      {
        \cs_gset_eq:NN \__LuaCJK_fv_use_verbatim: \FV@UseVerbatim
        \cs_gset_protected:Npn \FV@UseVerbatim ##1
          {
            \group_begin:
              \bool_set_true:N \l__LuaCJK_verbatim_block_bool
              \__LuaCJK_fv_use_verbatim: {##1}
            \group_end:
          }
      }
  }
\hook_gput_code:nnn { package / fancyvrb / after } { LuaCJK }
  { \__LuaCJK_fv_patch: }
\__LuaCJK_fv_patch:
```

(*End of definition for* `\__LuaCJK_verbatim_block_disable:`, `\__LuaCJK_fv_prep_hook:`, *and* `\__LuaCJK_fv_patch:`.)

`\__LuaCJK_verbatim_env_begin:`
`\__LuaCJK_verbatim_env_end:`

Environments that do not go through `\verbatim@font` need their own hooks. Only the `true` setting suspends the adjustments here: `env` and `env+` keep them, which is what keeps code columns aligned when CJK text is used inside the code. The begin hook marks the block for `\__LuaCJK_fv_prep_hook:` as well, so that the formatting hook of fancyvrb applies the rule of the environment rather than

the rule of \verb inside it.

```
\cs_new_protected:Npn \__LuaCJK_verbatim_env_begin:
  {
    \bool_set_true:N \l__LuaCJK_verbatim_block_bool
    \__LuaCJK_verbatim_block_disable:
  }
\cs_new_protected:Npn \__LuaCJK_verbatim_env_end:
  {
    \tl_if_eq:VnT \l__LuaCJK_verb_mode_tl { true }
      { \LuaCJKenable }
  }
\clist_map_inline:nn
  {
    verbatim,  Verbatim,  BVerbatim,
    verbatim*, Verbatim*, LVerbatim, lstlisting
  }
  {
    \hook_gput_code:nnn { env / #1 / begin } { LuaCJK }
      { \__LuaCJK_verbatim_env_begin: }
    \hook_gput_code:nnn { env / #1 / end   } { LuaCJK }
      { \__LuaCJK_verbatim_env_end:   }
  }
```

(*End of definition for* `\__LuaCJK_verbatim_env_begin:` *and* `\__LuaCJK_verbatim_env_end:`.)

## A.3 Fonts

`\__LuaCJK_family_font:n` Choose the font of a CJK family without selecting it. fontspec declares a family under the Unicode encoding, so a lookup made in another encoding finds no such shape, warns that the shape is undefined, and falls back to the Latin font: with fontenc's `T1` option the CJK text would be set in the T1 text font, and inside pifont's `U` encoding in the symbol font. Every place that resolves the font identifier of a family goes through this command, and `\selectfont` stays with the caller, which also sets the series, the shape, and the size where it needs them.

```
\cs_new_protected:Npn \__LuaCJK_family_font:n #1
  { \fontencoding { TU } \fontfamily {#1} }
```

(*End of definition for* `\__LuaCJK_family_font:n`.)

`\__LuaCJK_set_family:nnn` Define a family through fontspec and remember the resulting family name. The family is only registered; the switch happens in `\__LuaCJK_switch_family:n`. The registered families, and the family name currently selected.

`\g__LuaCJK_family_prop`
`\l__LuaCJK_family_tl`

```
\prop_new:N \g__LuaCJK_family_prop
\tl_new:N \l__LuaCJK_family_tl
```

```
\tl_set:Nn \l__LuaCJK_family_tl { zhsong }
```

Scratch variables holding the generated fontspec family name, the family name looked up in the property list, and a temporary value.

\l__LuaCJK_fontspec_tl
\l__LuaCJK_family_name_tl
\l__LuaCJK_tmpa_tl

```
\tl_new:N \l__LuaCJK_fontspec_tl
\tl_new:N \l__LuaCJK_family_name_tl
\tl_new:N \l__LuaCJK_tmpa_tl
\tl_new:N \l__LuaCJK_tmpb_tl
\int_new:N \l__LuaCJK_tmpa_int
\tl_new:N \l__LuaCJK_registered_name_tl
```

The font name and the explicit options of each family, kept so that `\addCJKfontfeatures` can register the family again, and the features prepended to every family. Whether one family's options already mention `AutoFakeBold`, `AutoFakeSlant`, `BoldFont`, or `ItalicFont`.

\g__LuaCJK_font_name_prop
\g__LuaCJK_font_feature_prop
\l__LuaCJK_default_feature_tl
\l__LuaCJK_autofake_bold_given_bool
\l__LuaCJK_autofake_slant_given_bool
\l__LuaCJK_bold_font_given_bool
\l__LuaCJK_italic_font_given_bool

```
\bool_new:N \l__LuaCJK_autofake_bold_given_bool
\bool_new:N \l__LuaCJK_autofake_slant_given_bool
\bool_new:N \l__LuaCJK_bold_font_given_bool
\bool_new:N \l__LuaCJK_italic_font_given_bool
\prop_new:N \g__LuaCJK_font_name_prop
\prop_new:N \g__LuaCJK_font_feature_prop
\tl_new:N  \l__LuaCJK_default_feature_tl
\tl_set:Nn \l__LuaCJK_default_feature_tl { Script = CJK }
\tl_new:N  \l__LuaCJK_feature_tl
\cs_generate_variant:Nn \fontspec_gset_family:Nnn { NVV , NVn }

\cs_new_protected:Npn \__LuaCJK_set_family:nnn #1#2#3
  {
    \__LuaCJK_build_features:n {#2}
    \prop_gput:NnV \g__LuaCJK_font_feature_prop {#1} \l__LuaCJK_feature_tl
    \prop_gput:Nnn \g__LuaCJK_font_name_prop {#1} {#3}
    \__LuaCJK_register_family:n {#1}
  }
```

Combine the default features, the options given for this family, and the automatic fake bold and slant, unless the family sets them itself.

__LuaCJK_build_features:n

```
\cs_new_protected:Npn \__LuaCJK_note_feature_key:nn #1#2
  { \__LuaCJK_note_feature_key:n {#1} }
\cs_new_protected:Npn \__LuaCJK_note_feature_key:n #1
  {
    \str_case:en { \tl_trim_spaces:n {#1} }
      {
        { AutoFakeBold  } { \bool_set_true:N \l__LuaCJK_autofake_bold_given_bool  }
        { AutoFakeSlant } { \bool_set_true:N \l__LuaCJK_autofake_slant_given_bool }
```

```
        { BoldFont      } { \bool_set_true:N \l__LuaCJK_bold_font_given_bool      }
        { ItalicFont    } { \bool_set_true:N \l__LuaCJK_italic_font_given_bool    }
      }
  }
\cs_new_protected:Npn \__LuaCJK_build_features:n #1
  {
    \clist_map_inline:nn
      { autofake_bold, autofake_slant, bold_font, italic_font }
      { \exp_args:Nc \bool_set_false:N { l__LuaCJK_##1_given_bool } }
    \keyval_parse:NNn
      \__LuaCJK_note_feature_key:n \__LuaCJK_note_feature_key:nn {#1}
    \tl_set_eq:NN \l__LuaCJK_feature_tl \l__LuaCJK_default_feature_tl
    \tl_if_blank:nF {#1}
      { \tl_put_right:Nn \l__LuaCJK_feature_tl { , ~ #1 } }
    \tl_if_empty:NF \g__LuaCJK_autofake_bold_tl
      {
        \bool_lazy_or:nnF
          { \l__LuaCJK_autofake_bold_given_bool }
          { \l__LuaCJK_bold_font_given_bool     }
          {
            \tl_put_right:Nx \l__LuaCJK_feature_tl
              { , ~ AutoFakeBold = \exp_not:V \g__LuaCJK_autofake_bold_tl }
          }
      }
    \tl_if_empty:NF \g__LuaCJK_autofake_slant_tl
      {
        \bool_lazy_or:nnF
          { \l__LuaCJK_autofake_slant_given_bool }
          { \l__LuaCJK_italic_font_given_bool    }
          {
            \tl_put_right:Nx \l__LuaCJK_feature_tl
              { , ~ AutoFakeSlant = \exp_not:V \g__LuaCJK_autofake_slant_tl }
          }
      }
  }
```

Build the fontspec family from the stored font name and features, and re-select it if it is the current one.

__LuaCJK_register_family:n

```
\cs_new_protected:Npn \__LuaCJK_register_family:n #1
  {
    \prop_get:NnNF \g__LuaCJK_font_name_prop {#1} \l__LuaCJK_registered_name_tl
      { \tl_clear:N \l__LuaCJK_registered_name_tl }
    \prop_get:NnNF \g__LuaCJK_font_feature_prop {#1} \l__LuaCJK_feature_tl
      { \tl_clear:N \l__LuaCJK_feature_tl }
    \prop_get:NnNF \g__LuaCJK_shift_decl_prop {#1} \l__LuaCJK_shift_dim_tl
```

```
      { \tl_clear:N \l__LuaCJK_shift_dim_tl }
    \group_begin:
      \fontspec_gset_family:NVV \l__LuaCJK_fontspec_tl
        \l__LuaCJK_feature_tl \l__LuaCJK_registered_name_tl
      \prop_gput:NnV \g__LuaCJK_family_prop {#1} \l__LuaCJK_fontspec_tl
      \tl_if_empty:NF \l__LuaCJK_shift_dim_tl
        {
          \tl_set:Ne \l__LuaCJK_shift_key_tl { #1 | \l__LuaCJK_fontspec_tl }
          \prop_gput:NVV \g__LuaCJK_font_shift_prop
            \l__LuaCJK_shift_key_tl \l__LuaCJK_shift_dim_tl
        }
    \group_end:
    \prop_if_in:NVT \g__LuaCJK_family_prop \l__LuaCJK_family_tl
      { \selectfont }
  }
```

__LuaCJK_push_fallback:
__LuaCJK_refresh_fallbacks:

Resolve the fallback families of the current CJK family to font identifiers at the current size, series, and shape, and hand the list to Lua together with the identifier of the family itself. A font identifier belongs to one instance of a font, so the list has to be resolved again whenever a font is selected: the identifier of the bold or enlarged main font is not the one the upright size was resolved at. The CJK family whose list was last handed to Lua for each font identifier, which makes a selection that resolves to a known identifier cost one property lookup, and whether the Lua module has been loaded. The guard of `\__LuaCJK_update_font:` is raised around the resolution, so the temporary selections it makes do not change the identifier they are resolved for. The math chain is refreshed in the same pass, so a fallback declared after the document started, or an `AutoFallBack` that is switched on, reaches the next formula even when no font is selected before it.

```
\prop_new:N \g__LuaCJK_fallback_pushed_prop
\bool_new:N \g__LuaCJK_lua_loaded_bool
\cs_new_protected:Npn \__LuaCJK_push_fallback:
  {
    \int_compare:nNnT { \g__LuaCJK_fontid_int } > { 0 }
      {
        \tl_set:Ne \l__LuaCJK_fontid_tl { \int_use:N \g__LuaCJK_fontid_int }
        \prop_get:NVNT \g__LuaCJK_fallback_prop \l__LuaCJK_family_tl \l__LuaCJK_tmpa_t
          {
            \prop_get:NVNF \g__LuaCJK_fallback_pushed_prop
              \l__LuaCJK_fontid_tl \l__LuaCJK_tmpb_tl
              { \tl_clear:N \l__LuaCJK_tmpb_tl }
            \tl_if_eq:NNF \l__LuaCJK_tmpb_tl \l__LuaCJK_family_tl
              { \exp_args:NV \__LuaCJK_push_fallback_aux:n \l__LuaCJK_tmpa_tl }
```

```
          }
      }
  }
\cs_new_protected:Npn \__LuaCJK_push_fallback_aux:n #1
  {
    \group_begin:
      \tl_clear:N \l__LuaCJK_fallback_ids_tl
      \clist_map_inline:nn {#1}
        {
          \__LuaCJK_family_font:n {##1} \selectfont
          \tl_set:Ne \l__LuaCJK_tmpb_tl
            { \int_eval:n { \tex_fontid:D \tex_font:D } }
          \tl_put_right:NV \l__LuaCJK_fallback_ids_tl \l__LuaCJK_tmpb_tl
          \tl_put_right:Nn \l__LuaCJK_fallback_ids_tl { , }
          \__LuaCJK_push_shift:nnn { \l__LuaCJK_tmpb_tl } { \l__LuaCJK_family_tl } {##
        }
      \bool_if:NT \g__LuaCJK_lua_loaded_bool
        {
          \tl_if_empty:NF \l__LuaCJK_fallback_ids_tl
            {
              \lua_now:e
                {
                  LuaCJK.set_fallback
                    ( \int_use:N \g__LuaCJK_fontid_int ,
                      { \l__LuaCJK_fallback_ids_tl } )
                }
              \prop_gput:NVV \g__LuaCJK_fallback_pushed_prop
                \l__LuaCJK_fontid_tl \l__LuaCJK_family_tl
            }
        }
    \group_end:
  }
\cs_new_protected:Npn \__LuaCJK_refresh_fallbacks:
  {
    \prop_gclear:N \g__LuaCJK_fallback_pushed_prop
    \bool_set_true:N \l__LuaCJK_selectfont_bool
    \__LuaCJK_push_fallback:
    \__LuaCJK_math_font_update:
    \bool_set_false:N \l__LuaCJK_selectfont_bool
  }
```

__LuaCJK_push_shift:nnn

Hand the vertical offset of one font to Lua. It is called for the current family, for each of its fallback fonts, and for the family `PunctFamily` names. One font instance carries one offset, so a paragraph is set with the value in force when it is shaped: the entry is keyed by the declaring family as well as by the family

fontspec returned, and a value is handed over again when it differs from the one Lua holds, which is what lets a fallback font and a family that name the same font take turns. A change also drops the record of which fallback list was handed over for a font identifier, because a list that shares the instance has to be resolved again for its own value. The `Baseline` value of each declared font, keyed by the declaring family and the family fontspec returned for it; the value Lua holds for each font identifier, and scratch variables for the identifier, the offset, and the two values that decide whether it has to be handed over, the offset being evaluated at the size in force so that a value in `em` follows the text.

```
\g__LuaCJK_font_identifier_prop
\g__LuaCJK_shift_pushed_prop
\l__LuaCJK_shift_dim_tl
\l__LuaCJK_shift_fid_tl
\l__LuaCJK_shift_key_tl
\l__LuaCJK_shift_value_tl
\l__LuaCJK_shift_old_tl
1032  \cs_new_protected:Npn \__LuaCJK_push_shift:nnn #1#2#3
1033    {
1034      \tl_set:Ne \l__LuaCJK_shift_fid_tl {#1}
1035      \tl_set:Ne \l__LuaCJK_shift_key_tl { #2 | #3 }
1036      \prop_get:NVNF \g__LuaCJK_font_shift_prop \l__LuaCJK_shift_key_tl
1037        \l__LuaCJK_shift_dim_tl
1038        { \tl_clear:N \l__LuaCJK_shift_dim_tl }
1039      \tl_if_empty:NTF \l__LuaCJK_shift_dim_tl
1040        { \tl_set:Nn \l__LuaCJK_shift_value_tl { 0 } }
1041        {
1042          \tl_set:Ne \l__LuaCJK_shift_value_tl
1043            {
1044              \dim_to_decimal_in_sp:n
1045                { \dim_eval:n { \l__LuaCJK_shift_dim_tl } }
1046            }
1047        }
1048      \prop_get:NVNF \g__LuaCJK_shift_pushed_prop \l__LuaCJK_shift_fid_tl
1049        \l__LuaCJK_shift_old_tl
1050        { \tl_set:Nn \l__LuaCJK_shift_old_tl { 0 } }
1051      \tl_if_eq:NNF \l__LuaCJK_shift_value_tl \l__LuaCJK_shift_old_tl
1052        {
1053          \prop_gput:NVV \g__LuaCJK_shift_pushed_prop
1054            \l__LuaCJK_shift_fid_tl \l__LuaCJK_shift_value_tl
1055          \prop_gclear:N \g__LuaCJK_fallback_pushed_prop
1056          \lua_now:e
1057            {
1058              LuaCJK.set_yoffset
1059                ( \l__LuaCJK_shift_fid_tl , \l__LuaCJK_shift_value_tl )
1060            }
1061        }
1062    }
```

(*End of definition for* `\__LuaCJK_set_family:nnn` *and others.*)

__LuaCJK_switch_family:n Select a registered family and warn once about an unknown name.

```
\cs_new_protected:Npn \__LuaCJK_switch_family:n #1
  {
    \prop_if_in:NnTF \g__LuaCJK_family_prop {#1}
      {
        \tl_set:Nn \l__LuaCJK_family_tl {#1}
        \selectfont
      }
      {
        \seq_if_in:NnF \g__LuaCJK_unknown_family_seq {#1}
          {
            \seq_gput_right:Nn \g__LuaCJK_unknown_family_seq {#1}
            \msg_warning:nnn { LuaCJK } { family-unknown } {#1}
          }
      }
  }
\seq_new:N \g__LuaCJK_unknown_family_seq
\msg_new:nnn { LuaCJK } { family-unknown }
  { Unknown ~ CJK ~ family ~ `#1' ~ is ~ being ~ ignored. }
```

(*End of definition for* `\__LuaCJK_switch_family:n`.)

__LuaCJK_reset_family: Reset the CJK family to `\CJKfamilydefault`. The name is expanded first, because the default is written as `\CJKrmdefault` until the document chooses a family of its own. A name that no family carries is passed over in silence: `\normalfont` also runs when the document starts, and the hook at `\begin{document}` runs the same reset once the default has been settled, so a document that declared no font at all has nothing to switch to.

```
\cs_new_protected:Npn \__LuaCJK_reset_family:
  {
    \tl_set:Ne \l__LuaCJK_tmpa_tl { \CJKfamilydefault }
    \prop_if_in:NVT \g__LuaCJK_family_prop \l__LuaCJK_tmpa_tl
      { \exp_args:NV \__LuaCJK_switch_family:n \l__LuaCJK_tmpa_tl }
  }
```

(*End of definition for* `\__LuaCJK_reset_family:`.)

__LuaCJK_ensure_family_default: Check that the family `\CJKfamilydefault` names was declared. A document that declares only some of its families sets its main text in no CJK font at all, and says nothing about it, so the family is replaced by `\CJKrmdefault` when that one is declared and by the first declared family otherwise. A document that declared nothing gets a warning of its own, because `LoadFandol` is off and there is no family to fall back to.

```
\cs_new_protected:Npn \__LuaCJK_ensure_family_default:
```

```
  {
    \tl_set:Ne \l__LuaCJK_tmpa_tl { \CJKfamilydefault }
    \prop_if_in:NVF \g__LuaCJK_family_prop \l__LuaCJK_tmpa_tl
      {
        \prop_if_empty:NTF \g__LuaCJK_family_prop
          { \msg_warning:nn { LuaCJK } { no-cjk-font } }
          {
            \tl_set:Ne \l__LuaCJK_tmpb_tl { \CJKrmdefault }
            \prop_if_in:NVF \g__LuaCJK_family_prop \l__LuaCJK_tmpb_tl
              {
                \prop_map_inline:Nn \g__LuaCJK_family_prop
                  { \tl_set:Nn \l__LuaCJK_tmpb_tl {##1} \prop_map_break: }
              }
            \msg_warning:nnee { LuaCJK } { family-default-undefined }
              { \l__LuaCJK_tmpa_tl } { \l__LuaCJK_tmpb_tl }
            \tl_gset_eq:NN \CJKfamilydefault \l__LuaCJK_tmpb_tl
            \exp_args:NV \__LuaCJK_switch_family:n \l__LuaCJK_tmpb_tl
          }
      }
  }
\msg_new:nnn { LuaCJK } { no-cjk-font }
  {
    No ~ CJK ~ font ~ is ~ set, ~ and ~ LoadFandol ~ is ~ off.\\
    Declare ~ one ~ with ~ \token_to_str:N \setCJKmainfont.
  }
\msg_new:nnn { LuaCJK } { family-default-undefined }
  {
    The ~ default ~ CJK ~ family ~ `#1' ~ is ~ not ~ declared;\\
    using ~ `#2' ~ instead.
  }
```

(*End of definition for* `\__LuaCJK_ensure_family_default:`.)

`__LuaCJK_math_font_update:` Register the CJK family used by the current settings as a math family and hand the fallback chain of that family to Lua, or clear the math font attribute when `CJKmath` is off or the family is not known. `\__LuaCJK_update_font:` runs it at every font selection, and `\__LuaCJK_math_version_refresh:` runs it again when the math version changes, which is why it stands on its own.

```
\cs_new_protected:Npn \__LuaCJK_math_font_update:
  {
    \tl_if_eq:NnTF \l__LuaCJK_cjk_math_tl { 1 }
      {
        \tl_if_empty:NTF \l__LuaCJK_math_family_tl
          { \tl_set_eq:NN \l__LuaCJK_tmpa_tl \l__LuaCJK_family_tl }
          { \tl_set_eq:NN \l__LuaCJK_tmpa_tl \l__LuaCJK_math_family_tl }
```

```
        \prop_get:NVNTF \g__LuaCJK_family_prop \l__LuaCJK_tmpa_tl \l__LuaCJK_tmpb_tl
          {
            \exp_args:NV \__LuaCJK_register_math_family:n \l__LuaCJK_tmpb_tl
            \exp_args:NV \__LuaCJK_math_fallbacks_push:n \l__LuaCJK_tmpa_tl
          }
          { \g__LuaCJK_math_font_attr = \c_zero_int }
      }
      { \g__LuaCJK_math_font_attr = \c_zero_int }
  }
```

(*End of definition for* `\__LuaCJK_math_font_update:`.)

`\__LuaCJK_update_font:` On every `\selectfont`, resolve the current family, its fallback families, the math family, and the `PunctFamily` family to font identifiers and store them in the attributes, and hand the `Baseline` offset of each to Lua. The punctuation family takes a selection of its own, because nothing reaches it through `\CJKfamily`; without the offset handed over here its marks would stand on the baseline of a font they are not set in. A guard against re-entering the update while `\selectfont` runs, and the font identifier of the current CJK family.

`\l__LuaCJK_selectfont_bool`
`\g__LuaCJK_fontid_int`

```
\bool_new:N \l__LuaCJK_selectfont_bool
\int_new:N \g__LuaCJK_fontid_int
```

The math family registered by `\setCJKmathfont`, the family chosen by `PunctFamily`, and the font identifier resolved for the latter.

`\l__LuaCJK_math_family_tl`
`\l__LuaCJK_punct_family_tl`
`\g__LuaCJK_punct_fontid_int`

```
\tl_new:N \l__LuaCJK_math_family_tl
\tl_new:N \l__LuaCJK_punct_family_tl
\int_new:N \g__LuaCJK_punct_fontid_int
```

The last math family number handed out, the map from a CJK family name to its number, and the upper bound. A CJK character in a math list needs a math family of its own, because the family is what selects the font *and* its size for the three math styles. LaTeX $2_\varepsilon$ allocates the 256 families that LuaTeX allows from zero upwards, so the numbers start at 240; a document that needs more than sixteen numbers is out of luck and gets an error. A fallback font used in math takes a number of its own, and the families that declare it share that number.

`\g__LuaCJK_math_family_int`
`\g__LuaCJK_math_family_prop`
`\c__LuaCJK_math_family_max_int`

```
\int_new:N \g__LuaCJK_math_family_int
\int_gset:Nn \g__LuaCJK_math_family_int { 239 }
\int_const:Nn \c__LuaCJK_math_family_max_int { 255 }
\prop_new:N \g__LuaCJK_math_family_prop

\cs_new_protected:Npn \__LuaCJK_update_font:
  {
    \bool_if:NF \l__LuaCJK_selectfont_bool
      {
```

```
      \bool_set_true:N \l__LuaCJK_selectfont_bool
      \prop_get:NVNTF \g__LuaCJK_family_prop \l__LuaCJK_family_tl \l__LuaCJK_family_
        {
          \group_begin:
            \exp_args:NV \__LuaCJK_family_font:n \l__LuaCJK_family_name_tl
            \selectfont
            \int_gset:Nn \g__LuaCJK_fontid_int
              { \tex_fontid:D \tex_font:D }
            \__LuaCJK_push_shift:nnn
              { \int_use:N \g__LuaCJK_fontid_int }
              { \l__LuaCJK_family_tl } { \l__LuaCJK_family_name_tl }
          \group_end:
          \g__LuaCJK_font_attr = \g__LuaCJK_fontid_int
        }
        {
          \g__LuaCJK_font_attr = \c_zero_int
          \tl_clear:N \l__LuaCJK_family_name_tl
        }
      \__LuaCJK_math_font_update:
      \__LuaCJK_push_fallback:
      \__LuaCJK_push_block_fonts:
      \tl_if_empty:NTF \l__LuaCJK_punct_family_tl
        { \g__LuaCJK_punct_font_attr = \c_zero_int }
        {
          \prop_get:NVNTF \g__LuaCJK_family_prop \l__LuaCJK_punct_family_tl
            \l__LuaCJK_tmpa_tl
            {
              \group_begin:
                \exp_args:NV \__LuaCJK_family_font:n \l__LuaCJK_tmpa_tl
                \selectfont
                \int_gset:Nn \g__LuaCJK_punct_fontid_int
                  { \tex_fontid:D \tex_font:D }
                \__LuaCJK_push_shift:nnn
                  { \int_use:N \g__LuaCJK_punct_fontid_int }
                  { \l__LuaCJK_punct_family_tl } { \l__LuaCJK_tmpa_tl }
              \group_end:
              \g__LuaCJK_punct_font_attr = \g__LuaCJK_punct_fontid_int
            }
            { \g__LuaCJK_punct_font_attr = \c_zero_int }
        }
      \bool_set_false:N \l__LuaCJK_selectfont_bool
    }
}
```

(*End of definition for* `\__LuaCJK_update_font:` *and others.*)

__LuaCJK_math_family:n

Give a CJK family or one of its fallback fonts a math family number. A number is handed out once per family name, so two CJK families that declare the same fallback font share its number.

```
\msg_new:nnn { LuaCJK } { math-family-exhausted }
  {
    Too ~ many ~ CJK ~ and ~ fallback ~ families ~ for ~ the ~ math ~
    families ~ 240--255.
  }
\cs_new_protected:Npn \__LuaCJK_math_family:n #1
  {
    \prop_if_in:NnF \g__LuaCJK_math_family_prop {#1}
      {
        \int_gincr:N \g__LuaCJK_math_family_int
        \int_compare:nNnT \g__LuaCJK_math_family_int > \c__LuaCJK_math_family_max_int
          { \msg_error:nn { LuaCJK } { math-family-exhausted } }
        \prop_gput:NnV \g__LuaCJK_math_family_prop {#1}
          \g__LuaCJK_math_family_int
      }
  }
```

(*End of definition for* `\__LuaCJK_math_family:n`.)

__LuaCJK_math_size_set:
\l__LuaCJK_math_size_tl

The text size a math family is built at: the size LaTeX $2_\varepsilon$ has installed the math fonts for, which is `\glb@currsize`, and `\f@size` before the document starts. The two differ inside amsmath's `\text`, whose `\mathchoice` sets `\f@size` to the script and scriptscript size for the branches it discards, and a math family built there would keep that size for every character after the `\text`.

```
\tl_new:N \l__LuaCJK_math_size_tl
\cs_new_protected:Npn \__LuaCJK_math_size_set:
  {
    \tl_set_eq:NN \l__LuaCJK_math_size_tl \f@size
    \cs_if_exist:NT \glb@currsize
      {
        \tl_if_empty:NF \glb@currsize
          { \tl_set_eq:NN \l__LuaCJK_math_size_tl \glb@currsize }
      }
  }
```

(*End of definition for* `\__LuaCJK_math_size_set:` *and* `\l__LuaCJK_math_size_tl`.)

__LuaCJK_math_family_fonts:nn

Fill the three style fonts of one math family number from a CJK family, at the text size of the formula and at seven and five tenths of it, which is the ratio of the standard LaTeX size tables. The family is selected inside a group, so the caller's font is not disturbed, and the assignments are global because a math font belongs

to a family, not to a group. The series and the shape come from the math version, not from the text around the formula: the family is selected at `\mddefault` and `\shapedefault`, and, in the `bold` math version, at `\bfdefault` as well when the family declares that shape. The regular face is what loads the shape list the test reads, and a family without a bold face keeps its regular one in silence. A formula therefore follows `\boldmath`, and a `\bfseries` that happens to precede it no longer leaks into it. The font identifier of the text size is recorded in `\g__LuaCJK_math_textfont_int`, which is the font a fallback chain tests for the character, because coverage does not depend on the size.

```
\int_new:N \g__LuaCJK_math_textfont_int
\cs_new_protected:Npn \__LuaCJK_math_family_fonts:nn #1#2
  {
    \group_begin:
      \__LuaCJK_family_font:n {#1}
      \fontseries { \mddefault }
      \fontshape { \shapedefault }
      \fontsize { \l__LuaCJK_math_size_tl pt } { \baselineskip }
      \selectfont
      \bool_lazy_and:nnT
        { \str_if_eq_p:ee { \math@version } { bold } }
        {
          \cs_if_exist_p:c
            { \use:c { f@encoding } / #1 / \bfdefault / \shapedefault }
        }
        { \fontseries { \bfdefault } \selectfont }
      \global \textfont #2 = \font
      \int_gset:Nn \g__LuaCJK_math_textfont_int
        { \tex_fontid:D \tex_font:D }
      \fontsize
        { \dim_eval:n { \l__LuaCJK_math_size_tl pt * 7 / 10 } }
        { \baselineskip }
      \selectfont
      \global \scriptfont #2 = \font
      \fontsize
        { \dim_eval:n { \l__LuaCJK_math_size_tl pt * 5 / 10 } }
        { \baselineskip }
      \selectfont
      \global \scriptscriptfont #2 = \font
    \group_end:
  }
```

(*End of definition for* `\__LuaCJK_math_family_fonts:nn`.)

`\__LuaCJK_register_math_family:n` Register a CJK family as a math family and point the math font attribute at its

number. A CJK character in a math list needs a family rather than a box, because the family is what selects the font *and* its size for text, script, and scriptscript style; a box is built once and keeps the size it was built at.

```
\cs_new_protected:Npn \__LuaCJK_register_math_family:n #1
  {
    \__LuaCJK_math_family:n {#1}
    \int_set:Nn \l__LuaCJK_tmpa_int
      { \prop_item:Nn \g__LuaCJK_math_family_prop {#1} }
    \__LuaCJK_math_size_set:
    \__LuaCJK_math_family_fonts:nn {#1} \l__LuaCJK_tmpa_int
    \g__LuaCJK_math_font_attr = \l__LuaCJK_tmpa_int
  }
```

(*End of definition for* `\__LuaCJK_register_math_family:n`.)

`\__LuaCJK_math_fallbacks_push:n` Register the fallback fonts of a CJK family as math families and hand the chain to Lua, which sets the character in the first font that has it. The chain holds the text font of the family itself first, so that Lua can test it too, then each fallback in the order it was declared, as pairs of a font identifier and a math family number. The family is given by the key it was declared under, which is the key of the fallback list, so a formula falls back like the text of the same family.

```
\tl_new:N \l__LuaCJK_math_fallback_ids_tl
\cs_new_protected:Npn \__LuaCJK_math_fallbacks_push:n #1
  {
    \bool_if:NT \g__LuaCJK_lua_loaded_bool
      {
        \__LuaCJK_math_size_set:
        \tl_clear:N \l__LuaCJK_math_fallback_ids_tl
        \tl_put_right:Nx \l__LuaCJK_math_fallback_ids_tl
          {
            \int_use:N \g__LuaCJK_math_textfont_int ,
            \int_use:N \g__LuaCJK_math_font_attr ,
          }
        \prop_get:NnNT \g__LuaCJK_fallback_prop {#1} \l__LuaCJK_tmpa_tl
          {
            \exp_args:NV \clist_map_inline:nn \l__LuaCJK_tmpa_tl
              {
                \__LuaCJK_math_family:n {##1}
                \int_set:Nn \l__LuaCJK_tmpa_int
                  { \prop_item:Nn \g__LuaCJK_math_family_prop {##1} }
                \__LuaCJK_math_family_fonts:nn {##1} \l__LuaCJK_tmpa_int
                \tl_put_right:Nx \l__LuaCJK_math_fallback_ids_tl
                  {
                    \int_use:N \g__LuaCJK_math_textfont_int ,
```

```
                    \int_use:N \l__LuaCJK_tmpa_int ,
                  }
              }
          }
        \lua_now:e
          {
            LuaCJK.set_math_fallback
              ( \int_use:N \g__LuaCJK_math_font_attr ,
                { \l__LuaCJK_math_fallback_ids_tl } )
          }
      }
  }
```

(*End of definition for* `\__LuaCJK_math_fallbacks_push:n`.)

`\__LuaCJK_math_version_refresh:` Register the math family again when the math version changes. LaTeX $2_\varepsilon$ knows the numbers of its own math families only, so `\mathversion` cannot reach the fonts of a CJK math family: the version is applied by hand. `\glb@settings` is where a version change and a text-size change are applied, and it runs `\every@math@size` afterwards, so that register is where the refresh belongs. It runs there also when the group that changed the version ends and the version in force before the group is applied again, which is what keeps `{\boldmath $中$}` local. The guard stops the font selections of the registration from running the whole update again.

```
\cs_new_protected:Npn \__LuaCJK_math_version_refresh:
  {
    \bool_if:NF \l__LuaCJK_selectfont_bool
      {
        \bool_set_true:N \l__LuaCJK_selectfont_bool
        \__LuaCJK_math_font_update:
        \bool_set_false:N \l__LuaCJK_selectfont_bool
      }
  }
```

(*End of definition for* `\__LuaCJK_math_version_refresh:`.)

`__LuaCJK_toks_put_right:Nn` Append tokens to a token register such as `\every@math@size`. The register holds tokens rather than a token list variable, and `\tl_put_right:Nn` would put the name of the register into its own value, so the assignment is written out with `\the`.

```
\cs_new_protected:Npn \__LuaCJK_toks_put_right:Nn #1#2
  { #1 \expandafter { \tex_the:D #1 #2 } }
\exp_args:Nc \__LuaCJK_toks_put_right:Nn { every@math@size }
  { \__LuaCJK_math_version_refresh: }
```

(*End of definition for* `\__LuaCJK_toks_put_right:Nn`.)

`__LuaCJK_push_block_fonts:` Resolve the block fonts of the current family at the current size and hand them to Lua, keyed by the font identifier of the family. Font identifiers depend on the size, so this runs again whenever a font is selected.

```
\int_new:N \g__LuaCJK_block_fontid_int
\cs_new_protected:Npn \__LuaCJK_push_block_fonts:
  {
    \int_compare:nNnT { \g__LuaCJK_fontid_int } > { 0 }
      {
        \clist_map_inline:Nn \g__LuaCJK_block_names_clist
          {
            \tl_set:Ne \l__LuaCJK_block_key_tl { \l__LuaCJK_family_tl | ##1 }
            \int_gzero:N \g__LuaCJK_block_fontid_int
            \prop_get:NVNT \g__LuaCJK_family_block_prop
              \l__LuaCJK_block_key_tl \l__LuaCJK_tmpa_tl
              {
                \group_begin:
                  \exp_args:NV \__LuaCJK_family_font:n \l__LuaCJK_tmpa_tl
                  \selectfont
                  \int_gset:Nn \g__LuaCJK_block_fontid_int
                    { \tex_fontid:D \tex_font:D }
                \group_end:
              }
            \lua_now:e
              {
                LuaCJK.set_block_font
                  ( \int_use:N \g__LuaCJK_fontid_int ,
                    "\lua_escape:e {##1}" ,
                    \int_use:N \g__LuaCJK_block_fontid_int )
              }
          }
      }
  }
```

(*End of definition for* `\__LuaCJK_push_block_fonts:`.)

The glues and the settings belong to the size that is being selected, but `\baselineskip` has not changed yet when the `selectfont` hook runs, so they are refreshed after the font selection. The guard keeps the temporary font selections inside `\__LuaCJK_update_font:` from doing the same twice.

```
\hook_gput_code:nnn { selectfont } { LuaCJK } { \__LuaCJK_update_font: }
\hook_gput_code:nnn { cmd / selectfont / after } { LuaCJK }
  {
```

```
    \bool_if:NF \l__LuaCJK_selectfont_bool
      {
        \__LuaCJK_refresh_glues:
        \__LuaCJK_settings_push:
      }
  }
```

\CJKrmdefault
\CJKsfdefault
\CJKttdefault
\CJKfamilydefault

The CJK families used by the standard font commands. The family hooks below re-select the CJK family whenever \rmfamily and its siblings run, and \normalfont selects \CJKfamilydefault, which follows \CJKrmdefault until the document gives it a value of its own. The value \CJKfamilydefault starts with, kept so that the hook at \begin{document} can tell an untouched default from a chosen one.

```
\clist_map_inline:nn { { rm } { zhsong } , { sf } { zhhei } , { tt } { zhfs } }
  {          \l__LuaCJK_family_default_init_tl
    \exp_args:Nc \tl_new:N { CJK \use_i:nn #1 default }
    \exp_args:Nc \tl_set:Nx { CJK \use_i:nn #1 default }
      { \use_ii:nn #1 }
  }
\tl_new:N \CJKfamilydefault
\tl_set:Nn \CJKfamilydefault { zhsong }
\tl_new:N \l__LuaCJK_family_default_init_tl
\tl_set_eq:NN \l__LuaCJK_family_default_init_tl \CJKfamilydefault
```

(*End of definition for* \CJKrmdefault *and others.*)

\defaultCJKfontfeatures

Set the features prepended to every CJK family, replacing the initial Script = CJK. Families registered earlier keep the value they were given; call \setCJKmainfont again to apply a new default.

```
\NewDocumentCommand \defaultCJKfontfeatures { m }
  { \tl_set:Nn \l__LuaCJK_default_feature_tl {#1} }
```

(*End of definition for* \defaultCJKfontfeatures*. This function is documented on page 29.*)

\addCJKfontfeatures

Add features to the current family for the rest of the group. The star form is accepted for compatibility; LuaCJK has no concept of blocks, so both forms act on the same font.

```
\NewDocumentCommand \addCJKfontfeatures { s o m }
  {
    \exp_args:NV \__LuaCJK_add_family_features:n
      \l__LuaCJK_family_tl {#3}
  }
```

Append features to the stored ones of a family and register it again.

__LuaCJK_add_family_features:n

```
\cs_new_protected:Npn \__LuaCJK_add_family_features:n #1#2
  {
    \prop_get:NnNTF \g__LuaCJK_font_feature_prop {#1} \l__LuaCJK_feature_tl
      {
        \tl_set:Ne \l__LuaCJK_feature_tl
          {
            \exp_not:V \l__LuaCJK_feature_tl
            \tl_if_blank:nF {#2} { , \exp_not:n {#2} }
          }
        \prop_get:NnNF \g__LuaCJK_font_name_prop {#1} \l__LuaCJK_registered_name_tl
          { \tl_clear:N \l__LuaCJK_registered_name_tl }
        \prop_get:NnNF \g__LuaCJK_shift_decl_prop {#1} \l__LuaCJK_shift_dim_tl
          { \tl_clear:N \l__LuaCJK_shift_dim_tl }
        \group_begin:
          \fontspec_gset_family:NVV \l__LuaCJK_fontspec_tl
            \l__LuaCJK_feature_tl \l__LuaCJK_registered_name_tl
          \tl_if_empty:NF \l__LuaCJK_shift_dim_tl
            {
              \tl_set:Ne \l__LuaCJK_shift_key_tl
                { #1 | \l__LuaCJK_fontspec_tl }
              \prop_gput:NVV \g__LuaCJK_font_shift_prop
                \l__LuaCJK_shift_key_tl \l__LuaCJK_shift_dim_tl
            }
        \group_end:
        \prop_put:NnV \g__LuaCJK_family_prop       {#1}
          \l__LuaCJK_fontspec_tl
        \prop_put:NnV \g__LuaCJK_font_feature_prop {#1}
          \l__LuaCJK_feature_tl
        \selectfont
      }
      { \msg_warning:nnn { LuaCJK } { family-unknown } {#1} }
  }
```

(*End of definition for* `\addCJKfontfeatures` *and* `\__LuaCJK_add_family_features:n`. *This function is documented on page 29.*)

\setCJKmainfont
\setCJKsansfont
\setCJKmonofont
\setCJKfamilyfont

The public font commands, all built on `\__LuaCJK_set_family_with_options:nnnn`. The family of `\setCJKmainfont` and its siblings is already the value of `\CJKrmdefault` and friends; `\setCJKfamilyfont` expands its argument so that the same name reaches either command.

```
\NewDocumentCommand \setCJKmainfont { o m o }
  {
    \exp_args:NV \__LuaCJK_set_family_with_options:nnnn
      \CJKrmdefault {#1} {#2} {#3}
```

```
  }
\NewDocumentCommand \setCJKsansfont { o m o }
  {
    \exp_args:NV \__LuaCJK_set_family_with_options:nnnn
      \CJKsfdefault {#1} {#2} {#3}
  }
\NewDocumentCommand \setCJKmonofont { o m o }
  {
    \exp_args:NV \__LuaCJK_set_family_with_options:nnnn
      \CJKttdefault {#1} {#2} {#3}
  }
\NewDocumentCommand \setCJKfamilyfont { m o m o }
  {
    \tl_set:Ne \l__LuaCJK_tmpa_tl {#1}
    \exp_args:NV \__LuaCJK_set_family_with_options:nnnn
      \l__LuaCJK_tmpa_tl {#2} {#3} {#4}
  }
\NewDocumentCommand \setCJKfallbackfamilyfont { m o m o }
  {
    \tl_set:Ne \l__LuaCJK_fallback_key_tl {#1}
    \tl_if_novalue:nTF {#2}
      {
        \tl_if_novalue:nTF {#4}
          { \__LuaCJK_add_fallback:nn {  } {#3} }
          { \__LuaCJK_add_fallback:nn {#4} {#3} }
      }
      { \__LuaCJK_add_fallback:nn {#2} {#3} }
  }
```

Accept the options before or after the font name.

```
\cs_new_protected:Npn \__LuaCJK_set_family_with_options:nnnn #1#2#3#4
  {
    \tl_if_novalue:nTF {#2}
      {
        \tl_if_novalue:nTF {#4}
          { \__LuaCJK_set_family_with_options_aux:nnn {#1} {  } {#3} }
          { \__LuaCJK_set_family_with_options_aux:nnn {#1} {#4} {#3} }
      }
      { \__LuaCJK_set_family_with_options_aux:nnn {#1} {#2} {#3} }
  }
\cs_generate_variant:Nn \__LuaCJK_set_family:nnn { nVn }
\cs_new_protected:Npn \__LuaCJK_set_family_with_options_aux:nnn #1#2#3
  {
    \__LuaCJK_extract_font_keys:nn {#1} {#2}
    \__LuaCJK_set_family:nVn {#1} \l__LuaCJK_features_tl {#3}
```

__LuaCJK_set_family_with_options:nnnn

```
}
```

Register one fallback font for a CJK family, keeping the order in which the fonts were declared.

__LuaCJK_add_fallback:nn

```
\cs_new_protected:Npn \__LuaCJK_add_fallback:nn #1#2
  {
    \__LuaCJK_parse_font_options:n {#1}
    \tl_if_empty:NT \l__LuaCJK_shift_tl
      { \tl_set:Nn \l__LuaCJK_features_tl {#1} }
    \group_begin:
      \exp_args:NV \__LuaCJK_build_features:n \l__LuaCJK_features_tl
      \fontspec_gset_family:NVn
        \l__LuaCJK_fallback_family_tl \l__LuaCJK_feature_tl {#2}
      \tl_if_empty:NF \l__LuaCJK_shift_tl
        {
          \tl_set:Ne \l__LuaCJK_shift_key_tl
            { \l__LuaCJK_fallback_key_tl | \l__LuaCJK_fallback_family_tl }
          \prop_gput:NVV \g__LuaCJK_font_shift_prop
            \l__LuaCJK_shift_key_tl \l__LuaCJK_shift_tl
        }
      \prop_get:NVNF \g__LuaCJK_fallback_prop
        \l__LuaCJK_fallback_key_tl \l__LuaCJK_tmpa_tl
        { \tl_clear:N \l__LuaCJK_tmpa_tl }
      \clist_put_right:NV \l__LuaCJK_tmpa_tl \l__LuaCJK_fallback_family_tl
      \prop_gput:NVV \g__LuaCJK_fallback_prop
        \l__LuaCJK_fallback_key_tl \l__LuaCJK_tmpa_tl
    \group_end:
    \__LuaCJK_refresh_fallbacks:
  }
```

(*End of definition for* `\setCJKmainfont` *and others. These functions are documented on page 25.*)

\CJKfamily

Switch to a registered family and gobble the following spaces. The name is expanded first, so that a family is named by the macro that holds it: `\CJKfamily {⟨\CJKsfdefault⟩}` selects the family that `\sffamily` selects, and the name an undeclared family is reported with is the expanded one.

```
\NewDocumentCommand \CJKfamily { m }
  {
    \tl_set:Ne \l__LuaCJK_tmpa_tl {#1}
    \exp_args:NV \__LuaCJK_switch_family:n \l__LuaCJK_tmpa_tl
    \tex_ignorespaces:D
  }
```

(*End of definition for* `\CJKfamily`*. This function is documented on page 25.*)

\newCJKfontfamily Define a control sequence that switches to a new family, and register the family under the same name.

```
\NewDocumentCommand \newCJKfontfamily { o m o m o }
  {
    \tl_set:Ne \l__LuaCJK_tmpa_tl
      { \IfNoValueTF {#1} { \cs_to_str:N #2 } {#1} }
    \cs_set_protected:Npx #2
      { \exp_not:N \__LuaCJK_switch_family:n { \exp_not:V \l__LuaCJK_tmpa_tl } }
    \exp_args:NV \__LuaCJK_set_family_with_options:nnnn
      \l__LuaCJK_tmpa_tl {#3} {#4} {#5}
  }
```

(*End of definition for* `\newCJKfontfamily`*. This function is documented on page 25.*)

\setCJKmathfont Register a family used for CJK characters in math mode and refresh the attributes at once.

```
\NewDocumentCommand \setCJKmathfont { o m o }
  {
    \__LuaCJK_set_family_with_options:nnnn { LuaCJK / math } {#1} {#2} {#3}
    \tl_set:Nn \l__LuaCJK_math_family_tl { LuaCJK / math }
    \__LuaCJK_update_font:
  }
```

(*End of definition for* `\setCJKmathfont`*. This function is documented on page 30.*)

\CJKfontspec Register a one-off family and switch to it immediately. Gives every call its own family name. \g__LuaCJK_fontspec_int

```
\int_new:N \g__LuaCJK_fontspec_int
```

```
\NewDocumentCommand \CJKfontspec { o m }
  {
    \int_gincr:N \g__LuaCJK_fontspec_int
    \tl_set:Ne \l__LuaCJK_tmpa_tl
      { LuaCJK / fontspec / \int_use:N \g__LuaCJK_fontspec_int }
    \exp_args:NV \__LuaCJK_set_family_with_options:nnnn
      \l__LuaCJK_tmpa_tl {#1} {#2} { }
    \exp_args:NV \__LuaCJK_switch_family:n \l__LuaCJK_tmpa_tl
    \tex_ignorespaces:D
  }
```

(*End of definition for* `\CJKfontspec` *and* `\g__LuaCJK_fontspec_int`*. This function is documented on page 26.*)

\songti
\heiti
\fangsong
\kaishu
\lishu
\youyuan

Shortcuts for the usual families, plus the family hooks of the standard font commands.

```
\cs_new_protected:Npn \songti   { \CJKfamily { zhsong } }
```

```
\cs_new_protected:Npn \heiti    { \CJKfamily { zhhei  } }
\cs_new_protected:Npn \fangsong { \CJKfamily { zhfs   } }
\cs_new_protected:Npn \kaishu   { \CJKfamily { zhkai  } }
\cs_new_protected:Npn \lishu    { \CJKfamily { zhli   } }
\cs_new_protected:Npn \youyuan  { \CJKfamily { zhyou  } }
\clist_map_inline:nn { rm, sf, tt }
  {
    \hook_gput_code:nnn { cmd / #1family / after } { LuaCJK }
      { \exp_args:Nv \__LuaCJK_switch_family:n { CJK#1default } }
  }
\hook_gput_code:nnn { cmd / normalfont / after } { LuaCJK }
  { \__LuaCJK_reset_family: }
```

(*End of definition for* `\songti` *and others. These functions are documented on page 26.*)

## A.4 Names and configuration

`\normalspacedchars` The characters in the list get no glue on either side, which is what one wants for slashes, backslashes, and hyphens inside Latin text. The list is kept on the Lua side as a set of code points, so the setting is global: a group does not restore the previous list. The list, initially `/`, `_`, `\backslash_str`, and `-`.

`\g__LuaCJK_normal_spaced_chars_tl`

```
\tl_new:N \g__LuaCJK_normal_spaced_chars_tl
\tl_gset:Nn \g__LuaCJK_normal_spaced_chars_tl
  { / , _ , \c_backslash_str , - }

\cs_new_protected:Npn \__LuaCJK_set_normal_spaced_chars:
  {
    \lua_now:e
      {
        LuaCJK.set_normal_spaced_chars
          ( "\lua_escape:e { \g__LuaCJK_normal_spaced_chars_tl } " )
      }
  }
\NewDocumentCommand \normalspacedchars { m }
  {
    \tl_gset:Nn \g__LuaCJK_normal_spaced_chars_tl {#1}
    \__LuaCJK_set_normal_spaced_chars:
  }
```

(*End of definition for* `\normalspacedchars` *and* `\g__LuaCJK_normal_spaced_chars_tl`*. This function is documented on page 31.*)

`\__LuaCJK_setup_format_chars:` A character that only controls the text and has no advance of its own needs care: the font loader replaces one that the font does not have by a space glyph and a negative kern, and the spacing pass then sees a Latin atom and surrounds it

with CJKecglue on both sides. There are two kinds. U+180E, U+200B, U+2060, and U+FEFF carry no information a font can use, so they are given category 9 and removed before the font loader runs. U+200C, U+200D, and the variation selectors are the ones a font shapes with: they are left alone here, and the pass that runs before shaping lets them follow the font of the character in front of them, so that what the font leaves of them can be dropped afterwards. A character produced by \char bypasses the category codes and is left alone: \char asks for the glyph itself, not for the character's typographic role. Without pre_shaping_filter there is no pass to tell the two kinds apart, so every character of the set is removed, which is what the package did before the shaping change.

```
\clist_const:Nn \c__LuaCJK_removed_chars_clist
  { "180E , "200B , "2060 , "FEFF }
\clist_const:Nn \c__LuaCJK_attached_chars_clist
  { "200C , "200D }
\cs_new_protected:Npn \__LuaCJK_setup_format_chars:
  {
    \clist_map_inline:Nn \c__LuaCJK_removed_chars_clist
      { \char_set_catcode_ignore:n {##1} }
    \bool_if:NF \g__LuaCJK_shaping_bool
      {
        \clist_map_inline:Nn \c__LuaCJK_attached_chars_clist
          { \char_set_catcode_ignore:n {##1} }
        \int_step_inline:nnn { "FE00 } { "FE0F }
          { \char_set_catcode_ignore:n {##1} }
        \int_step_inline:nnn { "E0100 } { "E01EF }
          { \char_set_catcode_ignore:n {##1} }
      }
  }
```

(*End of definition for* __LuaCJK_setup_format_chars:.)

__LuaCJK_load_cj: Read the CJ set from LineBreak.txt once. The key CJLineBreak calls this when it selects strict; a second call is cheap because the Lua side remembers that it has read the file.

```
\cs_new_protected:Npn \__LuaCJK_load_cj:
  { \lua_now:e { LuaCJK.load_cj() } }
```

(*End of definition for* __LuaCJK_load_cj:.)

_LuaCJK_load_local_config: Load the file chosen by the LocalConfig option. The key is read while the package is being loaded, but the file is only input at the end, so that it can use everything LuaCJK defines.

```
\cs_new_protected:Npn \__LuaCJK_load_local_config:
```

```
  {
    \tl_if_empty:NF \l__LuaCJK_local_config_tl
      {
        \exp_args:NV \__LuaCJK_load_config_file:n \l__LuaCJK_local_config_tl
      }
  }
\cs_new_protected:Npn \__LuaCJK_load_config_file:n #1
  {
    \file_if_exist:nTF {#1}
      { \file_input:n {#1} }
      { \msg_warning:nnn { LuaCJK } { config-missing } {#1} }
  }
\msg_new:nnn { LuaCJK } { config-missing }
  { The ~ configuration ~ file ~ `#1' ~ was ~ not ~ found. }
```

(*End of definition for* `\__LuaCJK_load_local_config:`.)

`\__LuaCJK_load_fandol:` When the `LoadFandol` option is on and the document did not declare any CJK font, fall back to the four Fandol faces. This runs at `\begin{document}` so that a family declared anywhere in the preamble still wins, and a document that declares some families but not the default one is reported by `\__LuaCJK_ensure_family_default:` instead. The roman family is declared twice, once as the main family and once under the name `zhsong`, and both declarations carry the same options, because a family name holds one set of features and the later declaration would otherwise drop what the earlier one gave. Fandol Song has no italic face, and emphasis in Chinese is set in Kai, so the italic shape of the roman family is Fandol Kai; the other three families have no italic face of their own.

```
\cs_new_protected:Npn \__LuaCJK_load_fandol:
  {
    \setCJKmainfont{FandolSong-Regular.otf}
      [ BoldFont = FandolSong-Bold.otf, ItalicFont = FandolKai-
Regular.otf ]
    \setCJKsansfont{FandolHei-Regular.otf}
      [ BoldFont = FandolHei-Bold.otf ]
    \setCJKmonofont{FandolFang-Regular.otf}
    \setCJKfamilyfont{zhsong}{FandolSong-Regular.otf}
      [ BoldFont = FandolSong-Bold.otf, ItalicFont = FandolKai-
Regular.otf ]
    \setCJKfamilyfont{zhhei}{FandolHei-Regular.otf}
      [ BoldFont = FandolHei-Bold.otf ]
    \setCJKfamilyfont{zhkai}{FandolKai-Regular.otf}
    \setCJKfamilyfont{zhfs}{FandolFang-Regular.otf}
  }
```

(*End of definition for* `\__LuaCJK_load_fandol:`.)

## A.5 The Lua module

The Lua module reads the attribute numbers and installs its callbacks. The current font is resolved once the document starts.

```
\lua_load_module:n { LuaCJK }
\lua_now:e
  {
    LuaCJK.attr_font        =  \int_use:N   \c__LuaCJK_font_attr_int ;
    LuaCJK.attr_cjk_space   =  \int_use:N   \c__LuaCJK_cjk_space_attr_int ;
    LuaCJK.attr_xecglue     =  \int_use:N   \c__LuaCJK_xecglue_attr_int ;
    LuaCJK.attr_inserted    =  \int_use:N   \c__LuaCJK_inserted_attr_int ;
    LuaCJK.attr_math_font   =  \int_use:N   \c__LuaCJK_math_font_attr_int ;
    LuaCJK.attr_punct_font  =  \int_use:N   \c__LuaCJK_punct_font_attr_int ;
    LuaCJK.attr_nobreak     =  \int_use:N   \c__LuaCJK_nobreak_attr_int ;
    LuaCJK.attr_settings    =  \int_use:N   \c__LuaCJK_settings_attr_int ;
    LuaCJK.attr_disabled    =  \int_use:N   \c__LuaCJK_disabled_attr_int ;
    LuaCJK.attr_attached    =  \int_use:N   \c__LuaCJK_attached_attr_int ;
    LuaCJK.attr_explicit_space
                            =  \int_use:N   \c__LuaCJK_explicit_space_attr_int ;
    LuaCJK.name_ccglue      = "\cs_to_str:N \l__LuaCJK_ccglue_skip" ;
    LuaCJK.name_ecglue      = "\cs_to_str:N \l__LuaCJK_ecglue_skip" ;
  }
\bool_gset_true:N \g__LuaCJK_lua_loaded_bool
```

`\__LuaCJK_detect_shaping:` Ask the module whether luaotfload offers the callback that runs before shaping, and record the answer for the two places that depend on it: the character setup below and the pass that `LuaCJK-core.lua` installs. Without the callback the package keeps its previous behavior in full, including the input removal of every zero-width character, so that a system never applies half of the change.

```
\bool_new:N \g__LuaCJK_shaping_bool
\cs_new_protected:Npn \__LuaCJK_detect_shaping:
  {
    \str_if_eq:eeT
      { \lua_now:e { tex.sprint(tostring(LuaCJK.shaping)) } }
      { true }
      { \bool_gset_true:N \g__LuaCJK_shaping_bool }
    \bool_if:NF \g__LuaCJK_shaping_bool
      { \msg_warning:nn { LuaCJK } { no-pre-shaping } }
  }
\msg_new:nnn { LuaCJK } { no-pre-shaping }
  {
    This~ luaotfload~ has~ no~ pre_shaping_filter,~ so~ the~ CJK~ font~ is~
    applied~ after~ the~ text~ is~ shaped:~ a~ combining~ mark~ will~ not~
    attach~ to~ its~ base,~ and~ the~ joiners~ and~ variation~ selectors~
```

```
    are~ removed~ at~ the~ input.
  }
```

(*End of definition for* `\__LuaCJK_detect_shaping:`.)

Read the package options now that every helper a key code calls exists and the Lua module is loaded, which `CJLineBreak` needs. The hook at `\begin{document}` then settles the family the document starts in: `\CJKfamilydefault` follows `\CJKrmdefault` when the preamble did not choose it, `LoadFandol` supplies the four Fandol faces when the preamble declared no family at all, a chosen default that no family carries is replaced by one that does, and the family that survives is selected, so a preamble that redefines `\CJKfamilydefault` sets the main text in that family from the first line instead of waiting for a `\normalfont`.

```
\ProcessKeyOptions [ LuaCJK / options ]
\__LuaCJK_settings_push:
\__LuaCJK_set_normal_spaced_chars:
\__LuaCJK_detect_shaping:
\__LuaCJK_setup_format_chars:
\hook_gput_code:nnn { begindocument } { LuaCJK }
  {
    \tl_if_eq:NNT \CJKfamilydefault \l__LuaCJK_family_default_init_tl
      {
        \group_begin:
        \tl_gset:Ne \CJKfamilydefault
          {
            \str_case:onF { \familydefault }
              {
                { \rmdefault } { \exp_not:N \CJKrmdefault }
                { \sfdefault } { \exp_not:N \CJKsfdefault }
                { \ttdefault } { \exp_not:N \CJKttdefault }
              }
              { \exp_not:N \CJKrmdefault }
          }
        \group_end:
      }
    \bool_if:NT \g__LuaCJK_load_fandol_bool
      {
        \prop_if_empty:NT \g__LuaCJK_family_prop
          { \__LuaCJK_load_fandol: }
      }
    \__LuaCJK_ensure_family_default:
    \__LuaCJK_reset_family:
    \__LuaCJK_update_font:
```

```
    \g__LuaCJK_font_attr = \g__LuaCJK_fontid_int
  }
\__LuaCJK_load_local_config:
\hook_gput_code:nnn { package / listings / after } { LuaCJK }
  { \RequirePackage { LuaCJK-listings } }
\cs_if_exist:cT { ver@listings.sty }
  { \RequirePackage { LuaCJK-listings } }
```

⟨/package⟩

The package ships two Lua modules. `LuaCJK.lua` only loads the core module; all node processing lives in `LuaCJK-core.lua`.

LuaCJK The table shared by the two modules. The core module adds its functions and constants to it.

```
⟨*lua⟩
LuaCJK = LuaCJK or {}

require("LuaCJK-core")
⟨/lua⟩
```

(*End of definition for* `LuaCJK`.)

The core module starts by creating the shared table and caching the node identifiers. The callbacks run on every list, so these lookups happen once.

```
⟨*lua-core⟩
LuaCJK = LuaCJK or {}

```

M Short names for the module table, the node types, and the node functions.

```
local M               = LuaCJK
local GLYPH           = node.id("glyph")
local GLUE            = node.id("glue")
local HLIST           = node.id("hlist")
local VLIST           = node.id("vlist")
local KERN            = node.id("kern")
local LIGATURE        = node.id("ligature")
local PENALTY         = node.id("penalty")
local MATH            = node.id("math")
local DISC            = node.id("disc")
local WHATSIT         = node.id("whatsit")
local MARK            = node.id("mark")
local INS             = node.id("ins")
local ADJUST          = node.id("adjust")
local LOCAL_PAR       = node.id("local_par")
```

GLYPH GLUE HLIST VLIST KERN LIGATURE PENALTY MATH DISC WHATSIT MARK INS ADJUST LOCAL_PAR SPACESKIP ITALICCORR new_node free_node has_attribute set_attribute

```
local SPACESKIP       = node.subtype("spaceskip") or 13
local ITALICCORR      = node.subtype("italiccorrection") or 3
local new_node        = node.new
local free_node       = node.free
local has_attribute   = node.has_attribute
local set_attribute   = node.set_attribute
local unset_attribute = node.unset_attribute

```

(*End of definition for* `M` *and others.*)

`LuaCJK.shaping` Whether luaotfload offers the callback that runs before a list is shaped. The TEX side reads this to decide how the zero-width characters are handled at the input, and the module uses it to choose between the substitution before shaping and the copy after it.

```
M.shaping = luatexbase.callbacktypes ~= nil
  and luatexbase.callbacktypes.pre_shaping_filter ~= nil
```

(*End of definition for* `LuaCJK.shaping`.)

`LuaCJK.attr_font` The attribute numbers. The TEX side fills them in after loading the module.

`LuaCJK.attr_cjk_space` `LuaCJK.attr_xecglue` `LuaCJK.attr_inserted` `LuaCJK.attr_math_font` `LuaCJK.attr_explicit_space`

```
for _, field in ipairs {
  "font",      "cjk_space",  "xecglue", "inserted",
  "math_font", "punct_font", "nobreak", "settings",
  "attached",  "explicit_space",
} do
  local key = "attr_" .. field
  M[key] = M[key] or nil
end

```

(*End of definition for* `LuaCJK.attr_font` *and others.*)

## A.6 Settings and data

Most options describe how text is set, not how the document is configured, so they have to travel with the characters. A snapshot of the current options is registered once and its number is stored in an attribute on the glyphs created afterwards; a group that ends before the paragraph does therefore still decides how its own text is set.

`LuaCJK.settings` `aCJK.settings_by_signature` `LuaCJK.attr_settings` The snapshots, an index from a signature to a snapshot so that repeated font selections reuse one entry, and the attribute number.

```
local DEFAULT_SETTINGS = {
```

```
  ccglue        = { width = 0, stretch = 0, shrink = 0 },
  ecglue        = { width = 0, stretch = 0, shrink = 0 },
  punct_style   = "quanjiao",
  kinsoku       = true,
  cj_strict     = false,
  latin_punct   = false,
  cjk_math      = false,
  punct_width   = false,
  bound_width   = false,
  check_single  = false,
  widow_penalty = 10000,
  allow_break   = false,
  auto_fallback = false,
  blocks_off    = {},
}

M.settings              = { DEFAULT_SETTINGS }
M.settings_by_signature = {}

```

(*End of definition for* `LuaCJK.settings`*,* `LuaCJK.settings_by_signature`*, and* `LuaCJK.attr_settings`.)

`LuaCJK.new_settings` Read the current glue registers, build a snapshot, and select it. The glue has to be read here rather than passed in, because that is the only place where the registers still hold the values of the moment. The stretch and shrink orders travel with the value, so a register such as `0pt plus 1filll` can still fill a line after the node has been built.

```
local function glue_spec (name)
  local skip = name and tex.skip[name]
  if skip then
    return { width         = skip.width,
             stretch       = skip.stretch,
             shrink        = skip.shrink,
             stretch_order = skip.stretch_order,
             shrink_order  = skip.shrink_order }
  end
  return { width = 0, stretch = 0, shrink = 0 }
end

function M.new_settings (style, kinsoku, cj_strict, latin_punct, cjk_math,
                         punct_width, bound_width, check_single, widow_penalty,
                         allow_break, auto_fallback, off_list)
  local off = {}
  for name in string.gmatch(off_list or "", "[^,]+") do
```

```
    local trimmed = string.match(name, "^%s*(.-)%s*$")
    if trimmed ~= "" then
      off[trimmed] = true
    end
  end
  local entry = {
    ccglue        = glue_spec(M.name_ccglue),
    ecglue        = glue_spec(M.name_ecglue),
    punct_style   = style,
    kinsoku       = kinsoku,
    cj_strict     = cj_strict,
    latin_punct   = latin_punct,
    cjk_math      = cjk_math,
    punct_width   = punct_width,
    bound_width   = bound_width,
    check_single  = check_single,
    widow_penalty = widow_penalty,
    allow_break   = allow_break,
    auto_fallback = auto_fallback,
    blocks_off    = off,
  }
  local signature = table.concat({
    entry.ccglue.width, entry.ccglue.stretch, entry.ccglue.shrink,
    entry.ccglue.stretch_order, entry.ccglue.shrink_order,
    entry.ecglue.width, entry.ecglue.stretch, entry.ecglue.shrink,
    entry.ecglue.stretch_order, entry.ecglue.shrink_order,
    style, tostring(kinsoku), tostring(cj_strict), tostring(latin_punct),
    tostring(cjk_math),
    tostring(punct_width), tostring(bound_width),
    tostring(check_single), widow_penalty,
    tostring(allow_break), tostring(auto_fallback), off_list,
  }, "|")
  local id = M.settings_by_signature[signature]
  if not id then
    id                                = #M.settings + 1
    M.settings[id]                    = entry
    M.settings_by_signature[signature] = id
  end
  if M.attr_settings then
    tex.setattribute(M.attr_settings, id)
  end
  return id
end
```

(*End of definition for* `LuaCJK.new_settings`.)

`settings_of` The snapshot of a node, or the defaults when it carries none.

```
local function settings_of (n)
  if M.attr_settings and n then
    local id = has_attribute(n, M.attr_settings)
    if id and id > 0 and M.settings[id] then
      return M.settings[id]
    end
  end
  return M.settings[1]
end

M.settings_of = settings_of

```

(*End of definition for* `settings_of`.)

`LuaCJK.cj` The characters whose `Line_Break` property is `CJ` (the Japanese small kana and
`LuaCJK.cj_count` the like), how many there are, and whether the file has been read. The set is read
`LuaCJK.cj_loaded` from `LineBreak.txt` of the unicode-data bundle only when `CJLineBreak` asks for `strict`, so a document that keeps the default never pays for the file.

```
M.cj        = {}
M.cj_count  = 0
M.cj_loaded = false

local function read_cj (data)
  local set, count = {}, 0
  for first, last, class in data:gmatch("(%x+)%.%.(%x+)%s*;%s*(%a+)") do
    if class == "CJ" then
      for code = tonumber(first, 16), tonumber(last, 16) do
        set[code] = true
      end
      count = count + 1
    end
  end
  for code, class in data:gmatch("(%x+)%s*;%s*(%a+)") do
    if class == "CJ" then
      set[tonumber(code, 16)] = true
      count = count + 1
    end
  end
  return set, count
end
```

```
1805
1806 function M.load_cj ()
1807   if M.cj_loaded then
1808     return M.cj_count
1809   end
1810   M.cj_loaded = true
1811   local path = kpse and kpse.find_file and kpse.find_file("LineBreak.txt")
1812   local file = path and io.open(path)
1813   if not file then
1814     texio.write_nl("term and log",
1815       "Package LuaCJK Warning: LineBreak.txt not found; "
1816       .. "CJLineBreak = strict has no effect.")
1817     return 0
1818   end
1819   local data = file:read("*a")
1820   file:close()
1821   M.cj, M.cj_count = read_cj(data)
1822   return M.cj_count
1823 end
1824
```

(*End of definition for* `LuaCJK.cj`, `LuaCJK.cj_count`, *and* `LuaCJK.cj_loaded`.)

The middle dot, the quotation marks, the horizontal bar, the two dot leader, the ellipses, and the hyphenation point are shared with Latin scripts. With `LatinPunct` on, the marks that key lists use the Latin font and are not compressed; the em dash is not in the table, because it stays CJK.

`LuaCJK.is_cjk` Test whether a character is treated as CJK. The table is the set of code points this package sets with the CJK class. It is a policy rather than a Unicode property: the punctuation that CJK text shares with Latin (the middle dot U+00B7, the quotation marks, the horizontal bar U+2015, the two dot leader U+2025, the ellipses U+2026 and U+22EF, and the hyphenation point U+2027) is listed here even though its script is Common, and the combining Bopomofo tone marks U+02EA and U+02EB are left out on purpose, because they attach to the preceding letter and a glue between the two would break the attachment. The third argument is the font of the glyph, and it decides the code points below U+0100: a font that luaotfload did not load as Unicode holds an encoding rather than characters, so the number that reaches the node is a position in that encoding. Slot 0xB7 of pifont's dingbat font would otherwise be read as the shared middle dot, and a CJK font that lacks it would drop the character; the accent `\r{u}`, slot 0xB7 of a T1 text font, would be read the same way. A code point above U+00FF cannot be a slot of an 8-bit font, so a CJK character typed in a legacy encoding still

takes the CJK font, and a Unicode font is read as characters throughout. The script `scripts/check-charclass.lua` compares the table with `Scripts.txt` and reports a code point whose script is Han, Hiragana, Katakana, Bopomofo, or Hangul but which the table does not list.

```
M.class_override = {}
M.punct_override = {}

local CJK_RANGES = {
  { 0x00B7,  0x00B7 },  -- middle dot, shared with Latin
  { 0x1100,  0x11FF },  -- Hangul Jamo
  { 0x2014,  0x2014 },  -- em dash, shared with Latin
  { 0x2015,  0x2015 },  -- horizontal bar, shared with Latin
  { 0x2018,  0x2019 },  -- single quotation marks
  { 0x201C,  0x201D },  -- double quotation marks
  { 0x2025,  0x2025 },  -- two dot leader, shared with Latin
  { 0x2026,  0x2026 },  -- horizontal ellipsis, shared with Latin
  { 0x2027,  0x2027 },  -- hyphenation point, shared with Latin
  { 0x22EF,  0x22EF },  -- midline ellipsis, shared with Latin
  { 0x2E80,  0x9FFF },  -- radicals, punctuation, kana, ideographs
  { 0xA960,  0xA97C },  -- Hangul Jamo Extended-A
  { 0xAC00,  0xD7C6 },  -- Hangul syllables and Jamo Extended-B
  { 0xD7CB,  0xD7FB },  -- Hangul Jamo Extended-B
  { 0xF900,  0xFAFF },  -- compatibility ideographs
  { 0xFE30,  0xFE4F },  -- compatibility forms
  { 0xFF00,  0xFFEF },  -- fullwidth forms
  { 0x16FE2, 0x16FE3 }, -- Old Chinese hook and iteration marks
  { 0x16FF0, 0x16FF6 }, -- Han, Vietnamese alternate reading marks
  { 0x1AFF0, 0x1AFF3 }, -- Kana Extended-B
  { 0x1AFF5, 0x1AFFB },
  { 0x1AFFD, 0x1AFFE },
  { 0x1B000, 0x1B128 }, -- Kana Supplement and Kana Extended-A
  { 0x1B132, 0x1B132 },
  { 0x1B150, 0x1B152 },
  { 0x1B155, 0x1B155 },
  { 0x1B164, 0x1B168 },
  { 0x1F200, 0x1F200 }, -- square hiragana hoka
  { 0x20000, 0x3FFFF }, -- ideographic planes
}

local function in_ranges (c, ranges)
  for i = 1, #ranges do
    if c >= ranges[i][1] and c <= ranges[i][2] then
      return true
    end
```

```
  end
  return false
end

local LATIN_PUNCT = {
  [0x00B7] = true, [0x2018] = true, [0x2019] = true,
  [0x201C] = true, [0x201D] = true, [0x2025] = true,
  [0x2026] = true, [0x2027] = true, [0x22EF] = true,
}

local SLOT_FONT = { }

local function is_slot (c, f)
  if c > 0xFF or not f then
    return false
  end
  local known = SLOT_FONT[f]
  if known == nil then
    local data = font.getfont(f)
    known = not (data and data.encodingbytes == 2)
    SLOT_FONT[f] = known
  end
  return known
end

local function is_cjk (c, s, f)
  local override = M.class_override[c]
  if override ~= nil then
    return override
  end
  if s and s.latin_punct and LATIN_PUNCT[c] then
    return false
  end
  return in_ranges(c, CJK_RANGES) and not is_slot(c, f)
end

M.is_cjk = is_cjk

```

(*End of definition for* `LuaCJK.is_cjk`.)

`LuaCJK.is_attached` Test whether a character attaches to the character in front of it instead of standing on its own: the two joiners and the variation selectors. A font shapes with these, and their own advance is nothing, so they follow the font of the character they attach to and take no part in the classes. The combining marks are not in this set:

a CJK one is in the CJK ranges and is shaped with its base, which is where its attachment comes from.

```
local ATTACHED = {
  [0x200C] = true, [0x200D] = true,
}

local function is_attached (c)
  return ATTACHED[c]
    or (c >= 0xFE00  and c <= 0xFE0F)
    or (c >= 0xE0100 and c <= 0xE01EF)
end

M.is_attached = is_attached

```

(*End of definition for* `LuaCJK.is_attached`.)

`PUNCT_CLASS` The punctuation LuaCJK knows about, grouped as opening, closing, middle, dash, and ellipsis. The group decides compression, line-breaking prohibitions, and hanging. The groups are the lists of xeCJK: its `OP` and `PR` are opening, its `CL`, `PO`, `EX`, and `IS` are closing, its hyphens are dashes, and its `NS` is read as closing or middle. A mark is listed here only when the CJK table holds it as well, so the small-form and vertical variants outside that table keep no class. The ASCII brackets are the exception, because they are set in the Latin font and cannot be compressed or hung: `half-open` and `half-close` are their own groups, which only the prohibitions and the spacing read.

```
local PUNCT_CLASS = {
  [0x0028] = "half-open",  [0x005B] = "half-open",
  [0x007B] = "half-open",
  [0x0029] = "half-close", [0x005D] = "half-close",
  [0x007D] = "half-close",
  [0x2018] = "open",     [0x201C] = "open",
  [0x3008] = "open",     [0x300A] = "open",
  [0x300C] = "open",     [0x300E] = "open",
  [0x3010] = "open",     [0x3014] = "open",
  [0x3016] = "open",     [0x3018] = "open",
  [0x301A] = "open",     [0x301D] = "open",
  [0xFE35] = "open",     [0xFE37] = "open",
  [0xFE39] = "open",     [0xFE3B] = "open",
  [0xFE3D] = "open",     [0xFE3F] = "open",
  [0xFE41] = "open",     [0xFE43] = "open",
  [0xFE47] = "open",     [0xFF04] = "open",
  [0xFF08] = "open",     [0xFF3B] = "open",
  [0xFF5B] = "open",     [0xFF5F] = "open",
```

```
  [0xFF62] = "open",     [0xFFE1] = "open",
  [0xFFE5] = "open",     [0xFFE6] = "open",
  [0x2019] = "close",    [0x201D] = "close",
  [0x3001] = "close",    [0x3002] = "close",
  [0x3009] = "close",    [0x300B] = "close",
  [0x300D] = "close",    [0x300F] = "close",
  [0x3011] = "close",    [0x3015] = "close",
  [0x3017] = "close",    [0x3019] = "close",
  [0x301B] = "close",    [0x301E] = "close",
  [0x301F] = "close",    [0xFE36] = "close",
  [0xFE38] = "close",    [0xFE3A] = "close",
  [0xFE3C] = "close",    [0xFE3E] = "close",
  [0xFE40] = "close",    [0xFE42] = "close",
  [0xFE44] = "close",    [0xFE48] = "close",
  [0xFF01] = "close",    [0xFF05] = "close",
  [0xFF09] = "close",    [0xFF0C] = "close",
  [0xFF0E] = "close",    [0xFF1A] = "close",
  [0xFF1B] = "close",    [0xFF1F] = "close",
  [0xFF3D] = "close",    [0xFF5D] = "close",
  [0xFF60] = "close",    [0xFF61] = "close",
  [0xFF63] = "close",    [0xFF64] = "close",
  [0xFFE0] = "close",
  [0x00B7] = "middle",   [0x2027] = "middle",
  [0x30FB] = "middle",   [0xFF65] = "middle",
  [0x2013] = "dash",     [0x2014] = "dash",
  [0x2015] = "dash",     [0x301C] = "dash",
  [0x30A0] = "dash",     [0xFF0D] = "dash",
  [0xFF5E] = "dash",
  [0x2025] = "ellipsis", [0x2026] = "ellipsis", [0x22EF] = "ellipsis",
}

```

(*End of definition for* `PUNCT_CLASS`.)

`half_mark` Whether a group belongs to the half-width brackets.

```
local function half_mark (mark)
  return mark == "half-open" or mark == "half-close"
end

```

(*End of definition for* `half_mark`.)

`punct_class` Look up the group of a character. The font is passed on to `\LuaCJK.is_cjk`'s rule, because a slot of a legacy font is a punctuation mark no more than it is a

CJK character: the middle dot of pifont's slot 0xB7 takes no compression and no hanging.

```
local function punct_class (c, s, f)
  local override = M.punct_override[c]
  if override ~= nil then
    return override or nil
  end
  if is_slot(c, f) then
    return nil
  end
  if s and s.latin_punct and LATIN_PUNCT[c] then
    return nil
  end
  if M.charsets.middle[c] then
    return "middle"
  end
  return PUNCT_CLASS[c]
end

```

(*End of definition for* `punct_class`.)

`LuaCJK.declare_ranges` Fill a table with ⟨*value*⟩ over a list of code-point ranges, each given as `{first, last}`.

```
local function fill_ranges (target, value, ranges)
  for _, range in ipairs(ranges) do
    local first = range[1]
    local last  = range[2] or first
    for code = first, last do
      target[code] = value
    end
  end
end

function M.declare_class (ranges, cjk)
  fill_ranges(M.class_override, cjk, ranges)
end

function M.declare_punct (ranges, group)
  fill_ranges(M.punct_override, group, ranges)
end

function M.reset_classes ()
  M.class_override = {}
  M.punct_override = {}
```

```
  M.normal_spaced  = {}
end
```

(*End of definition for* `LuaCJK.declare_ranges`.)

`M.charsets` The four character sets that the punctuation options can replace. `kaiming` holds the marks that keep a full em width in the `kaiming` style; `middle` and `long` are filled from the punctuation groups below, and `nobreak` starts as a copy of `long`.

```
M.charsets = {
  kaiming = {
    [0xFF0E] = true,  -- .
    [0x3002] = true,  -- 。
    [0xFF1F] = true,  -- ?
    [0xFF01] = true,  -- !
  },
  middle  = {},
  long    = {},
  nobreak = {},
}

for code, group in pairs(PUNCT_CLASS) do
  if group == "middle" then
    M.charsets.middle[code] = true
    PUNCT_CLASS[code]       = nil
  elseif group == "dash" or group == "ellipsis" then
    M.charsets.long[code] = true
  end
end

for code in pairs(M.charsets.long) do
  M.charsets.nobreak[code] = true
end
```

(*End of definition for* `M.charsets`.)

`LuaCJK.set_charset` Replace, extend, or reduce one of the sets above. The ranges arrive from TEX as a list of `{first, last}` pairs.

```
function M.set_charset (kind, mode, ranges)
  local target = M.charsets[kind]
  if not target then
    return
  end
  if mode == "set" then
```

```
    target           = {}
    M.charsets[kind] = target
  end
  for _, range in ipairs(ranges) do
    local first = range[1]
    local last  = range[2] or first
    for code = first, last do
      if mode == "remove" then
        target[code] = nil
      else
        target[code] = true
      end
    end
  end
end
```

(*End of definition for* `LuaCJK.set_charset`.)

`LuaCJK.punct_styles` The built-in styles. Each entry holds the compression switch, the compression ratio, the opening and closing exemption, whether the line-end hanging applies, and the optional fixed width.

```
M.punct_class = function (c, s, f)
  return punct_class(c, s or M.settings[1], f)
end

M.punct_styles = {
  quanjiao = { compress_adjacent = true,  ratio = 0.5,
               keep_open_close = true, hang = true },
  plain    = { compress_adjacent = false, ratio = 0.0,
               keep_open_close = true },
  banjiao  = { compress_adjacent = false, ratio = 0.0, keep_open_close = true,
               fixed_ratio = 0.5, hang = true },
  kaiming  = { compress_adjacent = false, ratio = 0.0, keep_open_close = true,
               fixed_ratio = 0.5, fixed_full = true, hang = true },
  CCT      = { compress_adjacent = false, ratio = 0.0, keep_open_close = true,
               fixed_ratio = 0.7, hang = true },
  hangmobanjiao = { compress_adjacent = false, ratio = 0.0,
                    keep_open_close = true, hang = true },
}
```

(*End of definition for* `LuaCJK.punct_styles`.)

`LuaCJK.punct_codes` The code points of the punctuation groups that lie inside the CJK ranges and the Basic Multilingual Plane, sorted. This is the list of marks that LuaCJK-listings

gives a token of their own. The filter is the CJK range table rather than a range list of its own, so a new Unicode release reaches the character table and this list together, and the groups are the ones LuaCJK.punct_class reads.

```
function M.punct_codes ()
  local codes = {}
  for code in pairs(PUNCT_CLASS) do
    codes[code] = true
  end
  for code in pairs(M.charsets.middle) do
    codes[code] = true
  end
  for code, group in pairs(M.punct_override) do
    codes[code] = group or nil
  end
  local list = {}
  for code in pairs(codes) do
    if code <= 0xFFFF and in_ranges(code, CJK_RANGES) then
      list[#list + 1] = code
    end
  end
  table.sort(list)
  return list
end
```

(*End of definition for* `LuaCJK.punct_codes`.)

`LuaCJK.define_punct_style` Register a style declared by `\LuaCJKDeclarePunctStyle`.

```
function M.define_punct_style (name, compress, ratio, keep)
  M.punct_styles[name] = {
    compress_adjacent = compress,
    ratio             = ratio,
    keep_open_close   = keep,
  }
end

```

(*End of definition for* `LuaCJK.define_punct_style`.)

`punct_style` Read the style name from the TEX token list and return the matching table, falling back to `plain`.

```
M.punct_width = {}
M.punct_kern  = {}

local function first_code (s)
  for _, code in utf8.codes(s) do
```

```
    return code
  end
end

function M.set_punct_width (chars, sp)
  for _, code in utf8.codes(chars) do
    M.punct_width[code] = sp
  end
end

function M.add_normal_spaced (ranges)
  for _, range in ipairs(ranges) do
    local first = range[1]
    local last  = range[2] or first
    for code = first, last do
      M.normal_spaced[code] = true
    end
  end
end

function M.set_punct_kern (before, after, sp)
  local a, b = first_code(before), first_code(after)
  if a and b then
    M.punct_kern[a]    = M.punct_kern[a] or {}
    M.punct_kern[a][b] = sp
  end
end

local function user_width (char)
  return M.punct_width[char]
end

local function user_kern (before, after)
  local row = M.punct_kern[before]
  return row and row[after]
end

function M.edit_punct_style (name, compress, ratio, keep)
  local style = M.punct_styles[name]
  if not style then
    return
  end
  if compress ~= nil then
    style.compress_adjacent = compress
  end
```

```
  if ratio ~= nil then
    style.ratio = ratio
  end
  if keep ~= nil then
    style.keep_open_close = keep
  end
end

local function style_of (settings)
  local name = settings and settings.punct_style
  return M.punct_styles[name] or M.punct_styles.plain
end

```

(*End of definition for* `punct_style`.)

`LuaCJK.attr_disabled` The attribute number of the suspension switch.

```
M.attr_disabled = M.attr_disabled or nil

```

(*End of definition for* `LuaCJK.attr_disabled`.)

`disabled` Test the suspension attribute on a node.

```
local function disabled (n)
  if not (M.attr_disabled and n) then
    return false
  end
  local v = has_attribute(n, M.attr_disabled)
  return v ~= nil and v ~= 0
end

```

(*End of definition for* `disabled`.)

## A.7 Node rewriting

A CJK glyph carries the identifier of the family that applies to it in an attribute. The pass below replaces it by a copy in that font, and takes a sub-block font or a fallback font instead when one applies.

`M.blocks` The declared sub-blocks in declaration order, and the font identifier of each block
`M.block_order` for each family. The order decides which block wins when two ranges overlap.
`M.block_fonts`

```
M.blocks      = {}
M.block_order = {}
M.block_fonts = {}
```

```
function M.declare_block (name, ranges)
  if not M.blocks[name] then
    table.insert(M.block_order, name)
  end
  M.blocks[name] = { ranges = ranges }
end

function M.set_block_font (main, name, fid)
  local row = M.block_fonts[main]
  if not row then
    row                 = {}
    M.block_fonts[main] = row
  end
  if fid and fid > 0 then
    row[name] = fid
  else
    row[name] = nil
  end
end

```

(*End of definition for* `M.blocks`, `M.block_order`, *and* `M.block_fonts`.)

`block_font` The font of the first block of ⟨*main*⟩ that contains ⟨*char*⟩ and is not canceled in ⟨*settings*⟩.

```
local function block_font (main, char, settings)
  local row = M.block_fonts[main]
  if not row then
    return nil
  end
  local off = settings and settings.blocks_off or {}
  for _, name in ipairs(M.block_order) do
    local block = M.blocks[name]
    if block and not off[name] and row[name] then
      for _, range in ipairs(block.ranges) do
        if char >= range[1] and char <= (range[2] or range[1]) then
          return row[name]
        end
      end
    end
  end
  return nil
end

```

(*End of definition for* `block_font`.)

`M.fallback` The fallback fonts of each family, as identifiers, in the order in which they were declared.

```
M.fallback = {}

function M.set_fallback (main, list)
  M.fallback[main] = list
end

```

(*End of definition for* `M.fallback`.)

`M.yoffset` The vertical offset of each font, in scaled points, positive upwards. The `Baseline` key of a font declaration fills it, and `pre_shaping` adds it when it moves a glyph into the font, so the offset belongs to the font the glyph is set in and not to the text that asks for it.

```
M.yoffset = {}

function M.set_yoffset (fid, sp)
  M.yoffset[fid] = sp
end

```

(*End of definition for* `M.yoffset`.)

`has_char` Whether ⟨*fid*⟩ has a glyph for ⟨*char*⟩. The answer is cached per font and code point, because the pass asks it once per glyph. The cache, keyed by the font identifier and the code point packed into one number.

```
local has_char_cache = {}

local function has_char (fid, char)
  local key    = fid * 0x200000 + char
  local cached = has_char_cache[key]
  if cached ~= nil then
    return cached
  end
  local f             = font.getfont(fid)
  local ok            = f and f.characters and f.characters[char] ~= nil
  has_char_cache[key] = ok
  return ok
end

```

(*End of definition for* `has_char` *and* `has_char_cache`.)

`fallback_font` The first fallback font of ⟨*fid*⟩ that has ⟨*char*⟩, or nothing when the feature is off or no fallback covers the character.

```
local function fallback_font (fid, char, settings)
  if not (settings and settings.auto_fallback) then
    return nil
  end
  local list = M.fallback[fid]
  if not list then
    return nil
  end
  for _, candidate in ipairs(list) do
    if has_char(candidate, char) then
      return candidate
    end
  end
end

```

(*End of definition for* `fallback_font`.)

`target_font` The font a CJK glyph takes: the family recorded on the glyph, then a sub-block font when one is declared for its block, then the punctuation family when the character is a mark it covers, then a fallback font when the font that applies does not have the character.

```
local function target_font (n)
  if not M.attr_font then
    return nil
  end
  local target = has_attribute(n, M.attr_font)
  if not (target and target > 0) then
    return nil
  end
  local settings = settings_of(n)
  local block    = block_font(target, n.char, settings)
  if block then
    target = block
  end
  if M.attr_punct_font and punct_class(n.char, nil, n.font) then
    local punct = has_attribute(n, M.attr_punct_font)
    if punct and punct > 0 then
      target = punct
    end
  end
  if not has_char(target, n.char) then
```

```
    target = fallback_font(target, n.char, settings) or target
  end
  return target
end
```

(*End of definition for* `target_font`.)

`placeholder` Whether a glyph is the one luaotfload substitutes for a character the font does not have: the box it draws, or the slot it leaves empty.

```
local function placeholder (c)
  return c == 0 or (c >= 0xF0000 and c <= 0x10FFFD)
end
```

(*End of definition for* `placeholder`.)

`pre_shaping` Set each CJK glyph in the font that sets it, before the list is shaped, and let a character that attaches to the one in front of it follow that font. The substitution comes before shaping because shaping is what attaches a mark to its base and what a variation selector and a joiner act on; a glyph copied into another font afterwards keeps the form of the font it was shaped in. A glyph already in its font is left alone, so the pass is idempotent across the two lists it sees, the paragraph and each line packed from it, and the `Baseline` shift is added once. The joiner or selector is marked, because shaping may leave a space glyph and a negative kern where the font had no use for the character, and `drop_orphans` cannot recognize them afterwards. A CJK character that its CJK font does not have is remembered, because the box luaotfload leaves for it is removed only where the package moved the character into that font: a document with no CJK font at all still shows the box, which is what tells the reader that nothing sets the text.

```
local missing_chars = {}

local function pre_shaping (head)
  local last = nil
  local cur  = head
  while cur do
    if cur.id == GLYPH then
      if is_attached(cur.char) then
        if M.attr_attached then
          set_attribute(cur, M.attr_attached, 1)
        end
        if last then
          cur.font = last.font
        end
```

```
      elseif is_cjk(cur.char, settings_of(cur), cur.font) then
        local target = target_font(cur)
        if target and target > 0 and target ~= cur.font then
          if not has_char(target, cur.char) then
            missing_chars[target * 0x200000 + cur.char] = true
          end
          local shift = M.yoffset[target]
          if shift then
            cur.yoffset = (cur.yoffset or 0) + shift
          end
          cur.font = target
        end
      end
      last = cur
    end
    cur = cur.next
  end
  return head
end

M.pre_shaping = pre_shaping

```

(*End of definition for* `pre_shaping`.)

`drop_orphans` Remove what shaping left of the two kinds of character that must not reach the spacing pass. A marked joiner or selector has done its work by now: the font either used it, in which case it is a ligature or a variant glyph and the node is gone, or it left a space glyph whose advance a negative kern cancels. Both the glyph and the kern are freed, so the text around it closes up and the two CJK characters on its sides take their glue. A CJK character that its CJK font does not have leaves the box glyph the font loader draws; it is freed as well, because the character was moved into a font that does not set it rather than left in the text font.

```
local function drop_orphans (head)
  local prev = nil
  local cur  = head
  while cur do
    local nxt  = cur.next
    local drop = false
    if cur.id == GLYPH and M.attr_attached
       and has_attribute(cur, M.attr_attached) ~= nil then
      drop = true
      if nxt and nxt.id == KERN and nxt.kern == -(cur.width or 0) then
```

```
        local after = nxt.next
        free_node(nxt)
        nxt = after
      end
    elseif cur.id == GLYPH and placeholder(cur.char)
       and prev and prev.id == GLYPH
       and missing_chars[prev.font * 0x200000 + prev.char] then
      drop = true
    end
    if drop then
      if prev then
        prev.next = nxt
      else
        head = nxt
      end
      free_node(cur)
      cur = nxt
    else
      prev = cur
      cur  = nxt
    end
  end
  return head
end

```

(*End of definition for* `drop_orphans`.)

`copy_glyph` Copy a glyph node, changing only its font. Only the fallback for a luaotfload without `pre_shaping_filter` uses it.

```
local function copy_glyph (n, fid)
  local g = new_node("glyph")
  g.font = fid
  for _, field in ipairs {
    "char",     "attr",    "lang",
    "left",     "right",   "uchyph",
    "xoffset",  "yoffset", "expansion_factor",
    "data",     "subtype",
  } do
    g[field] = n[field]
  end
  local shift = M.yoffset[fid]
  if shift then
    g.yoffset = (g.yoffset or 0) + shift
  end
```

```
  return g
end
```

(*End of definition for* `copy_glyph`.)

`rewrite_fonts` Replace each CJK glyph by a copy in the font its attribute names, after shaping. This is the fallback for a luaotfload without `pre_shaping_filter`; the package then behaves as it did before the substitution moved in front of shaping, and every zero-width character is removed at the input, so no attached character reaches this point. A glyph that the old font did not have may sit next to the placeholder luaotfload emitted for it; the placeholder is freed at the same time, because it would otherwise add an advance of its own.

```
local function rewrite_fonts (head)
  local prev = nil
  local cur  = head
  while cur do
    local nxt = cur.next
    if cur.id == GLYPH and is_cjk(cur.char, settings_of(cur), cur.font) then
      local target = target_font(cur)
      if target and target > 0 and target ~= cur.font then
        if nxt and nxt.id == GLYPH and nxt.font == cur.font
           and placeholder(nxt.char) then
          local after = nxt.next
          free_node(nxt)
          nxt = after
        end
        local g = copy_glyph(cur, target)
        g.next  = nxt
        if prev then
          prev.next = g
        else
          head = g
        end
        free_node(cur)
        cur = g
      end
    end
    prev = cur
    cur  = nxt
  end
  return head
end
```

(*End of definition for* `rewrite_fonts`.)

`class_of_char` Map a character to `CJK` or `LATIN`, with the font the glyph was set in, which `\LuaCJK.is_cjk` needs for the code points below U+0100.

```
local function class_of_char (c, s, f)
  if is_cjk(c, s, f) then
    return "CJK"
  end
  return "LATIN"
end
```

(*End of definition for* `class_of_char`.)

`class_of_node` Return the class at both edges of a node. A box carries the class of the node at its edge, so filling glue or a kern between the last character and the edge leaves the box with no class there; a box of no size is transparent, and formulas count as Latin. Whether a node carries neither text nor advance, so that the text reaches the edge past it.

`transparent_node`

```
local function transparent_node (n)
  return n.id == WHATSIT or n.id == MARK or n.id == INS
      or n.id == ADJUST or n.id == PENALTY
end
```

`box_visible` Whether a box occupies an advance of its own. A box that occupies none is passed over, so the boundary rules read the text on both sides of it as one run: an empty box does not break the run, and the ink of `\rlap`, `\llap` and `\clap` lies on the text around it instead of separating it.

```
local function box_visible (n)
  return n.width ~= 0
end
```

`attached_box` Whether a box belongs to the character in front of it. TEX writes the shift of a raised or lowered box into the box node, and `\textsuperscript` and `\textsubscript` build one, so the baseline of such a box is not the baseline of the text around it.

```
local function attached_box (n)
  return (n.id == HLIST or n.id == VLIST) and n.shift ~= 0
end
```

first_visible_class

The class of the text at the start of a list: a glyph, a formula, or a box inspected at its own edge. Glue, a kern, a rule, and a discretionary are not text, so a list that starts with one has no class at that point, and a box whose first edge has no class ends the search instead of yielding to a character after it.

```
local function first_visible_class (list)
  local n = list
  while n do
    if n.id == GLYPH then
      return class_of_char(n.char, settings_of(n), n.font)
    elseif n.id == HLIST or n.id == VLIST then
      return first_visible_class(n.head)
    elseif n.id == MATH then
      return "LATIN"
    elseif not transparent_node(n) then
      return nil
    end
    n = n.next
  end
  return nil
end

```

last_visible_class

The class of the text at the end of a list, read the same way: a node that is not text ends the text, so the class is nothing when something follows the last character.

```
local function last_visible_class (list)
  local n    = list
  local last = nil
  while n do
    if n.id == GLYPH then
      last = class_of_char(n.char, settings_of(n), n.font)
    elseif n.id == HLIST or n.id == VLIST then
      last = last_visible_class(n.head)
    elseif n.id == MATH then
      last = "LATIN"
    elseif not transparent_node(n) then
      last = nil
    end
    n = n.next
  end
  return last
end

```

first_punct

The punctuation group of the first visible character of a list, the character itself, and whether a character was found at all, so that a box with no text can

be skipped. A formula counts as material with no mark.

```
local function first_punct (list)
  local n = list
  while n do
    if n.id == GLYPH then
      return punct_class(n.char, settings_of(n), n.font), n, true
    elseif n.id == HLIST or n.id == VLIST then
      local p, g, found = first_punct(n.head)
      if found then
        return p, g, true
      end
    elseif n.id == MATH then
      return nil, nil, true
    end
    n = n.next
  end
  return nil, nil, false
end

```

The same for the last visible character of a list.

last_punct

```
local function last_punct (list)
  local n     = list
  local punct = nil
  local glyph = nil
  local found = false
  while n do
    if n.id == GLYPH then
      punct = punct_class(n.char, settings_of(n), n.font)
      glyph = n
      found = true
    elseif n.id == HLIST or n.id == VLIST then
      local p, g, deep = last_punct(n.head)
      if deep then
        punct = p
        glyph = g
        found = true
      end
    elseif n.id == MATH then
      punct = nil
      glyph = nil
      found = true
    end
    n = n.next
  end
```

```
  return punct, glyph, found
end

local function class_of_node (n)
  if n.id == GLYPH then
    local c = class_of_char(n.char, settings_of(n), n.font)
    return c, c
  elseif n.id == HLIST or n.id == VLIST then
    return first_visible_class(n.head), last_visible_class(n.head)
  elseif n.id == MATH then
    return "LATIN", "LATIN"
  end
  return nil, nil
end

```

(*End of definition for* `class_of_node` *and others.*)

`is_inserted` Test whether a node was inserted by LuaCJK.

```
local function is_inserted (n)
  return M.attr_inserted
    and has_attribute(n, M.attr_inserted) ~= nil
end

```

(*End of definition for* `is_inserted`.)

`inherit_attributes` Give a new node the attribute list of the node it is spliced next to. LuaCJK creates glue, kerns, and penalties after TEX has attached the surrounding attributes to the text, so a package that marks spans with attributes, such as lua-ul, would otherwise read the new node as the end of its span and start a new one. Assigning to the ⟨*attr*⟩ field is reference counted, and later per-attribute changes are copy-on-write, so the source node is never modified.

```
local function inherit_attributes (new, source)
  if source and source.attr then
    new.attr = source.attr
  end
end

```

(*End of definition for* `inherit_attributes`.)

`make_glue` Create a glue node from a skip register, inherit the attributes of ⟨*source*⟩, and mark the node as inserted. The node keeps the stretch and shrink orders of the register.

```
local function make_glue (spec, source)
```

```
  if not spec then
    return nil
  end
  local s = spec
  local g = new_node("glue")
  inherit_attributes(g, source)
  g.width         = s.width
  g.stretch       = s.stretch
  g.shrink        = s.shrink
  g.stretch_order = s.stretch_order or 0
  g.shrink_order  = s.shrink_order  or 0
  if M.attr_inserted then
    set_attribute(g, M.attr_inserted, 1)
  end
  return g
end

```

(*End of definition for* `make_glue`.)

`set_glue` Give an existing glue node the dimensions of a skip register, including its stretch and shrink orders.

```
local function set_glue (g, spec)
  if not spec then
    return false
  end
  local s   = spec
  g.width   = s.width
  g.stretch = s.stretch
  g.shrink  = s.shrink
  g.stretch_order = s.stretch_order or 0
  g.shrink_order  = s.shrink_order  or 0
  if M.attr_inserted then
    set_attribute(g, M.attr_inserted, 1)
  end
  return true
end

```

(*End of definition for* `set_glue`.)

`make_kern` Create a kern node, inherit the attributes of ⟨*source*⟩, and mark the node as inserted.

```
local function make_kern (width, source)
  local k = new_node("kern")
```

```
2568   inherit_attributes(k, source)
2569   k.kern = width
2570   if M.attr_inserted then
2571     set_attribute(k, M.attr_inserted, 1)
2572   end
2573   return k
2574 end
2575
```

(*End of definition for* `make_kern`.)

## A.8 The passes

`spacing` Walk a list and insert the glue between CJK and Latin material. The walk tracks the last visible node and the class at its end, so that a formula acts as one Latin atom at its left edge; inside the formula the boundary rules are not applied. A box is an atom as well, and the class at its edge is the class of the node at that edge: filling glue or a kern means the box has no text there, so no glue is added beside it. A box that occupies no advance is passed over, so an empty box inside CJK text does not break the run, and the ink of an overlaid one does not split it. A box whose own baseline TEX has shifted is attached to the character in front of it: the run ends at it, so it takes no glue on either side and offers no break. That is the box `\textsuperscript` and `\textsubscript` build; the shift is the box's own field, so the outer box of a `\raisebox` or of a footnote mark keeps its boundary. A penalty that prohibits the break is passed over the same way, because the glue of that boundary belongs after it; a smaller penalty, and a discretionary, end the run. Decide what to do at the boundary between two visible nodes. Only source-generated glue is a candidate for replacement; other glue is left alone, and a discretionary is opaque, so neither is bridged. A penalty is read by its value. One that prohibits the break is not a barrier: the glue goes after it, so the prohibition still holds at the break the glue offers. A smaller penalty prices the break instead of forbidding it, and the glue would offer the same place a second break at no cost, so a priced break is left alone. A source space between the two characters is the boundary itself, so when one stands in front of a prohibiting penalty it is converted, and the penalty moves in front of it rather than letting the space offer the break. A whatsit, a mark, an insertion, or an adjustment between that space and the character is passed over: a color change in front of the character does not hide the space, which stays the boundary and is never joined by a second glue. A penalty ends that search, because it has a boundary rule of its own. A boundary that touches a suspended run is bridged by nothing, because `Verb` removes the spacing; the break opportunity stays, as a glue of zero width in the

`handle_boundary`

place the spacing would have taken. The mark rules are applied by hand there, because the kinsoku pass cannot see a prohibition across a suspended node: no break follows an opening mark, none precedes a closing, middle, or long mark that begins a run, and two Latin words are never parted. Insert the CJK–CJK or CJK–Latin glue, or the compression kern for two adjacent punctuation marks. The character on each side is the one the list shows, so a whatsit or a penalty between two characters does not hide them from these rules; the glue itself follows the node that carries the break, which may be such a whatsit or penalty. A run of dashes or ellipses is one unit: the marks are joined without glue, so that justification cannot pull a two-em dash apart, and the absence of a break opportunity replaces the explicit prohibition. A punctuation mark takes no CJK–Latin advance on either side: the advance of the mark already carries its spacing, and the glue would add a third of an em that the typography does not ask for. The boundary keeps a glue of zero width, so that it stays a break opportunity and the kinsoku penalty can forbid the break. A half-width bracket is the exception: the boundary in front of an opening one keeps the spacing that the two classes give it, and the other three sides of the pair take no glue at all. The pair therefore hugs the text it encloses and offers no break inside, which is how xeCJK sets its half-width marks.

```
M.normal_spaced = {}

function M.set_normal_spaced_chars (chars)
  local set = {}
  for _, code in utf8.codes(chars) do
    set[code] = true
  end
  M.normal_spaced = set
end

local function normal_spaced (n)
  return n and n.id == GLYPH and M.normal_spaced[n.char] == true
end

local function long_punct (n)
  return n and n.id == GLYPH and M.charsets.long[n.char] == true
end

local PUNCT_BOUNDARY_GLUE = { width = 0, stretch = 0, shrink = 0 }

local function insert_boundary_glue (head, prev, prev_visible, prev_end,
                                     cur, cur_start, settings)
  if not prev_end or not cur_start then
```

```
    return head
  end
  if long_punct(prev_visible) and long_punct(cur) then
    return head
  end
  local style = style_of(settings)
  local prev_punct = nil
  if prev_visible then
    if prev_visible.id == GLYPH then
      prev_punct = punct_class(prev_visible.char, settings, prev_visible.font)
    elseif prev_visible.id == HLIST or prev_visible.id == VLIST then
      prev_punct = last_punct(prev_visible.head)
    end
  end
  local cur_punct = nil
  if cur then
    if cur.id == GLYPH then
      cur_punct = punct_class(cur.char, settings, cur.font)
    elseif cur.id == HLIST or cur.id == VLIST then
      cur_punct = first_punct(cur.head)
    end
  end
  local prev_mark = prev_punct and prev.id == GLYPH
      and not half_mark(prev_punct)
  local cur_mark  = cur_punct and cur.id == GLYPH
      and not half_mark(cur_punct)
  local spec = nil
  if prev_end == "CJK" and cur_start == "CJK" then
    spec = settings.ccglue
  elseif prev_end ~= cur_start
     and not normal_spaced(prev_visible) and not normal_spaced(cur) then
    if half_mark(prev_punct) then
      return head
    elseif cur_punct == "half-open" then
      spec = prev_punct and PUNCT_BOUNDARY_GLUE or settings.ecglue
    elseif half_mark(cur_punct) then
      return head
    elseif prev_punct or cur_punct then
      spec = PUNCT_BOUNDARY_GLUE
    else
      spec = settings.ecglue
    end
  end

  if prev_mark and cur_mark then
```

```
    local explicit = user_kern(prev.char, cur.char)
    if explicit then
      local k = make_kern(explicit, prev)
      k.next  = cur
      if prev then
        prev.next = k
      else
        head = k
      end
      return head
    end
  end
  if prev_mark and cur_mark and style.compress_adjacent then
    local exempt = (style.keep_open_close
        and prev_punct == "open" and cur_punct == "close")
      or (M.charsets.long[prev.char] and M.charsets.long[cur.char])
    if not exempt and prev.width and prev.width > 0 then
      local k = make_kern(-math.floor(style.ratio * prev.width + 0.5), prev)
      local g = spec and make_glue(spec, prev) or nil
      k.next  = g or cur
      if g then
        g.next = cur
      end
      if prev then
        prev.next = k
      else
        head = k
      end
      return head
    end
  end

  if not spec then
    return head
  end
  local g = make_glue(spec, prev or cur)
  if not g then
    return head
  end
  g.next = cur
  if prev then
    prev.next = g
  else
    head = g
  end
```

```
  return head
end
```

restore_source_spaces

Give a source space that a text font command moved across the italic correction its subtype back. `\textbf` and the other text font commands save the skip in front of them, remove it, apply the italic correction, and put it back with `\hskip`, so the glue keeps the dimensions of the space but not its subtype. The move is the only way a normal glue comes to follow an italic-correction kern, and the subtype is the one thing the spacing pass reads, so putting the subtype back is enough; a typeface whose interword space is rigid therefore needs no special case, at the price of retiming a skip that the document wrote in that very place. A space never has a stretch or shrink order, so a glue that has one was not a space at all: `\hfill` and `\hfil` follow the italic correction of the same command and keep their own subtype, or the fill of the line would become the inter-character glue.

```
local function restore_source_spaces (head)
  local prev = nil
  local cur  = head
  while cur do
    if cur.id == GLUE and cur.subtype ~= SPACESKIP
       and cur.stretch_order == 0 and cur.shrink_order == 0
       and prev and prev.id == KERN and prev.subtype == ITALICCORR then
      cur.subtype = SPACESKIP
    end
    prev = cur
    cur  = cur.next
  end
  return head
end
```

handle_source_space

Remove or retime a space typed in the source. Only a space that came from the source is a candidate; an explicit `\hskip` or `\hfill` has a different subtype and is left alone, and the space that a text font command moved across the italic correction has had its subtype put back above. A space between two CJK characters is not kept when `CJKspace` is off, but dropping it must not drop the boundary with it: the space becomes the inter-character glue of `CJKglue`. Freeing it instead would leave the two characters adjacent with no glue, and the pass that would add the glue runs after line breaking, so the break opportunity would be lost. A space the document wrote with `\ ` or `\nobreakspace` is not a source space, and the attribute set around it is the only place that says so: the two are the same `spaceskip` glue, with the same dimensions, until that mark is read. Such a space

keeps its advance, on either side of the boundary it stands at, because xeCJK sets it as a space too and CJKspace is about the source.

```
local function handle_source_space (head, glue, prev_end, cur,
                                    cur_start, settings)
  if glue.subtype ~= SPACESKIP then
    return head
  end
  if M.attr_explicit_space
     and (has_attribute(glue, M.attr_explicit_space) or 0) ~= 0 then
    return head
  end
  local prev_cjk = prev_end == "CJK"
  local cur_cjk  = cur_start == "CJK"
  if prev_cjk and cur_cjk then
    local cjk_space = M.attr_cjk_space
      and (has_attribute(glue, M.attr_cjk_space) or 0) ~= 0
    if not cjk_space then
      set_glue(glue, settings.ccglue)
      if M.attr_inserted then
        set_attribute(glue, M.attr_inserted, 1)
      end
    end
  elseif prev_cjk or cur_cjk then
    local xecglue = M.attr_xecglue
      and (has_attribute(glue, M.attr_xecglue) or 0) ~= 0
    if xecglue then
      set_glue(glue, settings.ecglue)
    end
  end
  return head
end

```

Whether the break beside a suspended run is allowed: the mark rules of `forbids_break` are read at the edge of the run, and two Latin words are left whole.

suspended_break

```
local function suspended_break (prev_visible, cur, prev_end, start_class,
                                settings)
  if not prev_end or not start_class then
    return false
  end
  if normal_spaced(prev_visible) or normal_spaced(cur) then
    return false
  end
```

```
  local prev_mark = prev_visible.id == GLYPH
    and punct_class(prev_visible.char, settings, prev_visible.font)
  local cur_mark = cur.id == GLYPH
    and punct_class(cur.char, settings, cur.font)
  if prev_mark == "open" or prev_mark == "half-open" then
    return false
  end
  if cur_mark == "close" or cur_mark == "middle"
     or cur_mark == "half-close" then
    return false
  end
  if (cur_mark == "dash" or cur_mark == "ellipsis")
     and not (prev_mark == "dash" or prev_mark == "ellipsis") then
    return false
  end
  if prev_end == "LATIN" and start_class == "LATIN"
     and not prev_mark and not cur_mark then
    return false
  end
  return true
end

local function handle_boundary (head, prev, prev_visible, prev_end,
                                cur, start_class, settings)
  if not prev_visible or not start_class then
    return head
  end
  if disabled(prev_visible) or disabled(cur) then
    local occupied = prev
      and (prev.id == GLUE or prev.id == KERN
           or prev.id == PENALTY or prev.id == DISC)
    if not occupied and disabled(prev_visible) ~= disabled(cur)
       and suspended_break(prev_visible, cur, prev_end, start_class, settings)
    then
      local source = disabled(cur) and prev_visible or cur
      local g = make_glue(PUNCT_BOUNDARY_GLUE, source)
      if g then
        g.next = cur
        if prev then
          prev.next = g
        else
          head = g
        end
      end
```

```
2789      end
2790      return head
2791    end
2792    if prev and prev.id == GLUE then
2793      if not is_inserted(prev) then
2794        return handle_source_space(
2795          head, prev, prev_end, cur, start_class, settings)
2796      end
2797    elseif prev and prev.id == PENALTY then
2798      if prev.penalty >= 10000 then
2799        local space, before = nil, nil
2800        local n = prev_visible
2801        while n and n ~= prev do
2802          local nxt = n.next
2803          if nxt and nxt.id == GLUE and nxt.subtype == SPACESKIP then
2804            space, before = nxt, n
2805            break
2806          end
2807          n = nxt
2808        end
2809        if space then
2810          head = handle_source_space(
2811            head, space, prev_end, cur, start_class, settings)
2812          local pred = before
2813          local m    = space
2814          while m and m ~= prev do
2815            pred = m
2816            m    = m.next
2817          end
2818          pred.next   = prev.next
2819          before.next = prev
2820          prev.next   = space
2821          return head
2822        end
2823        return insert_boundary_glue(
2824          head, prev, prev_visible, prev_end, cur, start_class, settings)
2825      end
2826    elseif prev and prev.id == DISC then
2827    elseif prev and transparent_node(prev) then
2828      local n = prev
2829      while n and n.id ~= PENALTY and transparent_node(n) do
2830        n = n.prev
2831      end
2832      if n and n.id == GLUE then
2833        if not is_inserted(n) then
```

```
        return handle_source_space(
          head, n, prev_end, cur, start_class, settings)
      end
      return head
    end
    return insert_boundary_glue(
      head, prev, prev_visible, prev_end, cur, start_class, settings)
  else
    return insert_boundary_glue(
      head, prev, prev_visible, prev_end, cur, start_class, settings)
  end
  return head
end

local function spacing (head)
  head = restore_source_spaces(head)
  local prev         = nil
  local prev_visible = nil
  local prev_end     = nil
  local math_level   = 0
  local cur          = head
  while cur do
    local nxt = cur.next
    if cur.id == MATH then
      if cur.subtype == 0 then
        if math_level == 0 then
          head = handle_boundary(
            head, prev, prev_visible, prev_end, cur, "LATIN",
            settings_of(prev_visible or cur))
        end
        math_level   = math_level + 1
        prev_visible = nil
        prev_end     = nil
      else
        if math_level > 0 then
          math_level = math_level - 1
        end
        if math_level == 0 then
          prev_visible = cur
          prev_end     = "LATIN"
        end
      end
    elseif math_level > 0 then
    else
```

```
      local start_class, end_class = class_of_node(cur)
      local visible
      if cur.id == HLIST or cur.id == VLIST then
        visible = box_visible(cur)
      else
        visible = start_class ~= nil
      end
      if visible and attached_box(cur) then
        prev_visible = nil
        prev_end     = nil
      elseif visible then
        head = handle_boundary(
          head, prev, prev_visible, prev_end, cur, start_class,
          settings_of(prev_visible))
        prev_visible = cur
        prev_end     = end_class
      elseif cur.id == PENALTY or cur.id == DISC then
        if not (cur.id == PENALTY and cur.penalty >= 10000) then
          prev_visible = nil
          prev_end     = nil
        end
      end
    end
    prev = cur
    cur  = nxt
  end
  return head
end
```

(*End of definition for* `spacing` *and others.*)

TEX may break at glue, at kern, and at a discretionary. The first two are places where LuaCJK inserts its own nodes, so after building the list a second pass puts a penalty 10000 in front of every break opportunity that must not be used. A discretionary states the value of its own break, so there the pass sets the node's penalty field rather than inserting a penalty. The hyphens of a word are not in the list yet when the pass runs: TEX inserts them during line breaking, so a `hyphenate` callback forbids those as well.

`apply_kinsoku` Add the prohibitions where the settings of the text ask for them. A node marked by `\LuaCJKnobreak` or by the verbatim addon is prohibited regardless, because the mark itself is the prohibition. The break points the pass reads are the glue LuaCJK inserts and the source spaces, because the punctuation rules are about

those places alone; the mark is about every break, so a glue that carries it is read as a break point as well, which is what keeps a `\quad` or an `\hspace` inside the argument on the same line. A box is read at the edge that faces the break, the way the spacing pass reads it, so that a mark at the edge of the box is neither a line start nor a line end.

Whether a node carries the attribute of `\LuaCJKnobreak`.

no_break_mark

```
local function no_break_mark (n)
  if not (M.attr_nobreak and n) then
    return false
  end
  local value = has_attribute(n, M.attr_nobreak)
  return value ~= nil and value ~= 0
end
```

The glyph a discretionary is followed by: the letters of a word are separated from it by a kern or a ligature at most, and a glue, a penalty, another discretionary or a formula ends the word.

next_glyph

```
local function next_glyph (n)
  while n and (n.id == KERN or n.id == LIGATURE) do
    n = n.next
  end
  if n and n.id == GLYPH then
    return n
  end
  return nil
end
```

Forbid the break at a discretionary that involves a marked node. The break lies between the letters on either side of the discretionary, so it is prohibited when either of them is marked, the way the break at a glue is. TEX gives a discretionary the attributes of the character in front of it, which is why the node itself is read as well as its neighbours.

forbid_marked_disc

```
local function forbid_marked_disc (disc, left)
  if no_break_mark(disc) or no_break_mark(left)
     or no_break_mark(next_glyph(disc.next)) then
    disc.penalty = 10000
  end
end
```

adjacent_marks

Whether two marks are neighbours that stay together: a dash or an ellipsis is left out, because a run of them is one unit of its own, governed by the `nobreak` character set.

```
local function adjacent_marks (prev_mark, next_mark)
  return prev_mark and next_mark
     and prev_mark ~= "dash" and prev_mark ~= "ellipsis"
     and next_mark ~= "dash" and next_mark ~= "ellipsis"
end

```

run_mark

Whether a punctuation mark is one of the two a run is built from.

```
local function run_mark (mark)
  return mark == "dash" or mark == "ellipsis"
end

```

forbids_break

The rules: closing, middle, dash, and ellipsis punctuation must not start a line, opening punctuation must not end one, a half-width bracket follows the rule of the side it closes, a run of dashes or ellipses stays together, and two adjacent marks are not separated. A dash or an ellipsis is read as the run it begins, so the rule applies to the first mark of a run alone; between the marks of a run the `NoBreakLongPunct` set decides. `AllowBreakBetweenPuncts` lifts the last rule alone: a mark that may not begin or end a line is a rule of the script, so the option cannot permit one.

```
local function forbids_break (prev_mark, prev_char, next_mark, next_char,
                              settings)
  if next_mark == "close" or next_mark == "middle"
     or next_mark == "half-close" then
    return true
  end
  if run_mark(next_mark) and not run_mark(prev_mark) then
    return true
  end
  if prev_mark == "open" or prev_mark == "half-open" then
    return true
  end
  if M.charsets.nobreak[prev_char] and M.charsets.nobreak[next_char] then
    return true
  end
  if adjacent_marks(prev_mark, next_mark)
     and not (settings and settings.allow_break) then
    return true
  end
  return false
```

```
end

```

edge_mark

The punctuation group and the character at one edge of a visible node: a glyph is its own edge, and a box is read at the edge that faces the break, so a mark inside a box takes part in the rules. A formula carries no mark.

```
local function edge_mark (n, at_end)
  if n.id == GLYPH then
    return punct_class(n.char, settings_of(n), n.font), n.char, n
  elseif n.id == HLIST or n.id == VLIST then
    local mark, g = nil, nil
    if at_end then
      mark, g = last_punct(n.head)
    else
      mark, g = first_punct(n.head)
    end
    if g then
      return mark, g.char, g
    end
    return nil, nil, nil
  end
  return nil, nil, nil
end

```

next_visible

The next node that carries text: a glyph, a formula, or a box. A box that occupies no advance is passed over, and an explicit break barrier ends the search.

```
local function next_visible (n)
  while n do
    if n.id == GLYPH or n.id == MATH then
      return n
    elseif n.id == HLIST or n.id == VLIST then
      if box_visible(n) then
        return n
      end
    elseif n.id == PENALTY or n.id == DISC then
      return nil
    end
    n = n.next
  end
  return nil
end

```

is_break_point

Glue and kerns are break opportunities; only the ones LuaCJK inserted and the source spaces are considered.

```
2998 local function is_break_point (n)
2999   if n.id == GLUE then
3000     return is_inserted(n) or n.subtype == SPACESKIP
3001   elseif n.id == KERN then
3002     return is_inserted(n)
3003   end
3004   return false
3005 end
3006
3007 local function apply_kinsoku (head)
3008   local prev = nil
3009   local last = nil
3010   local cur  = head
3011   while cur do
3012     if cur.id == GLYPH
3013        or ((cur.id == HLIST or cur.id == VLIST) and box_visible(cur)) then
3014       last = cur
3015     elseif cur.id == DISC then
3016       forbid_marked_disc(cur, last)
3017     elseif last and (is_break_point(cur)
3018                      or (cur.id == GLUE and no_break_mark(cur))) then
3019       local nxt                  = next_visible(cur.next)
3020       local settings             = settings_of(last)
3021       local prev_mark, prev_char = edge_mark(last, true)
3022       local next_mark, next_char = nil, nil
3023       if nxt then
3024         next_mark, next_char = edge_mark(nxt, false)
3025       end
3026       local cj = settings.cj_strict and next_char and M.cj[next_char]
3027       if nxt and (no_break_mark(cur) or no_break_mark(last)
3028                   or no_break_mark(nxt) or cj
3029                   or (settings.kinsoku
3030                       and not disabled(last) and not disabled(nxt)
3031                       and forbids_break(prev_mark, prev_char,
3032                                         next_mark, next_char, settings)))
3033       then
3034         if not (prev and prev.id == PENALTY and prev.penalty >= 10000) then
3035           local p = new_node("penalty")
3036           inherit_attributes(p, last)
3037           p.penalty = 10000
3038           if M.attr_inserted then
3039             set_attribute(p, M.attr_inserted, 1)
```

```
            end
            p.next = cur
            if prev then
              prev.next = p
            else
              head = p
            end
            cur = p
          end
        end
      end
      prev = cur
      cur  = cur.next
    end
    return head
end
```

(*End of definition for* `apply_kinsoku` *and others.*)

`apply_fixed_widths` Styles such as `banjiao` give every punctuation mark a fixed advance. The glyph advance cannot be changed, so a compensating kern goes right after the glyph. A penalty in front of that kern keeps TEX from treating it as an extra break opportunity. The kern alone is only right for a mark whose ink sits at the left of its advance, which is what a closing mark does. An opening mark keeps its ink at the right, so the ink is moved left by the same amount; a middle mark is centered, so it moves by half. A dash or an ellipsis is not shortened by the style: its advance is the length of the stroke, and halving it would let the two halves of a two-em dash overlap. A half-width bracket is already narrow, so the ratio of a style leaves it alone; a width the document sets applies to it like any other. A mark keeps the kern it was given. TEX runs the box-packing callback again for every box that holds the list, so a mark whose advance was corrected would be corrected a second time: the penalty and the kern the pass inserted are the record, and a mark that already carries them is passed over.

```
local function ink_shift (class, delta)
  if class == "open" or class == "half-open" then
    return delta
  elseif class == "middle" then
    return math.floor(delta / 2 + 0.5)
  end
  return 0
end
```

```
local function fixed_width (cur, settings, style)
  local target = user_width(cur.char)
  if not target and settings.punct_width then
    target = settings.punct_width
  end
  if target then
    return target
  end
  if half_mark(punct_class(cur.char, settings, cur.font)) then
    return nil
  end
  local ratio = style.fixed_ratio
  if not ratio or M.charsets.long[cur.char] then
    return nil
  end
  local full = style.fixed_full and M.charsets.kaiming
  if full and full[cur.char] then
    return nil
  end
  local f    = font.getfont(cur.font)
  local size = f and f.size or 0
  if size > 0 then
    return math.floor(ratio * size + 0.5)
  end
  return nil
end

local function fixed_done (n)
  local p = n.next
  if p and p.id == PENALTY and p.penalty == 10000 and is_inserted(p) then
    local k = p.next
    return k ~= nil and k.id == KERN and is_inserted(k)
  end
  return false
end

local function apply_fixed_widths (head)
  local cur = head
  while cur do
    local settings = settings_of(cur)
    local style    = style_of(settings)
    if cur.id == GLYPH and punct_class(cur.char, settings, cur.font)
       and not disabled(cur) and not fixed_done(cur) then
      local target = fixed_width(cur, settings, style)
      if target then
```

```
        local delta = target - cur.width
        if delta ~= 0 then
          local shift = ink_shift(punct_class(cur.char, settings, cur.font), delta)
          local p = new_node("penalty")
          inherit_attributes(p, cur)
          p.penalty = 10000
          if M.attr_inserted then
            set_attribute(p, M.attr_inserted, 1)
          end
          local k   = make_kern(delta, cur)
          p.next    = k
          k.next    = cur.next
          cur.next  = p
          if shift ~= 0 then
            cur.xoffset = cur.xoffset + shift
          end
        end
      end
    end
    cur = cur.next
  end
  return head
end

```

(*End of definition for* `apply_fixed_widths`.)

A CJK character in math mode reaches the math list as a `math_char` noad. Its family points at the math font, which usually has no CJK glyph, so LuaCJK replaces the nucleus by a `sub_box` holding the character in the CJK font.

`MATH_CHAR` `MATH_TEXT_CHAR` The node types of the two kinds of math character noad.

```
local MATH_CHAR      = node.id("math_char")
local MATH_TEXT_CHAR = node.id("math_text_char")

```

(*End of definition for* `MATH_CHAR` *and* `MATH_TEXT_CHAR`.)

`any_cjk_math_enabled` Whether any noad of the list asks for CJK in math mode. The switch is part of the settings of the text, so the answer is per list.

```
local function any_cjk_math_enabled (head)
  local cur = head
  while cur do
    if settings_of(cur).cjk_math then
      return true
```

```
    end
    cur = cur.next
  end
  return false
end
```

(*End of definition for* `any_cjk_math_enabled`.)

`SUB_MLIST` The node type of a nested math list.

```
local SUB_MLIST = node.id("sub_mlist")
```

(*End of definition for* `SUB_MLIST`.)

`rewrite_math` Rewrite the CJK noads of a math list, or warn once when the feature is off. Walk a math list, recursing into sub lists (fractions, braces, scripts) so that nested CJK characters are rewritten too. The declaration is split from the definition because of that recursion.

```
local rewrite_math_list
```

`M.math_fallback` The fallback chain of each CJK math family, as a flat list that alternates the identifier of the font a character is tested in and the number of the math family that sets it. The text font of the family itself comes first, so a character the family has is set by the family.

```
M.math_fallback = {}

function M.set_math_fallback (main, list)
  M.math_fallback[main] = list
end
```

`math_family_for` The first math family of the chain of ⟨*family*⟩ whose font has ⟨*char*⟩, or the family itself when none does. The fallback fonts are consulted only when `AutoFallBack` is on, which the settings of the noad carry, so the switch reaches math mode like it reaches the text.

```
local function math_family_for (char, family, settings)
  local list = M.math_fallback[family]
  if not list then
    return family
  end
  for i = 1, #list, 2 do
    if i > 1 and not (settings and settings.auto_fallback) then
```

```
      break
    end
    if has_char(list[i], char) then
      return list[i + 1]
    end
  end
  return family
end
```

Point the nucleus of one noad at the math family of its CJK family, or at the first fallback family that has the character, and recurse into the sub lists of its numerator, denominator, superscript, and subscript. The family number comes from the attribute TEX set for the current group, which is why `\setCJKmathfont` keeps its meaning inside a group: the family of the node changes, not a global font.

rewrite_noad

```
local function point_at_family (char, family)
  char.fam = family
  if M.attr_font then
    unset_attribute(char, M.attr_font)
  end
end

local function rewrite_field (s, noad, on, family)
  if not s then
    return
  end
  if s.id == MATH_CHAR or s.id == MATH_TEXT_CHAR then
    if is_cjk(s.char, settings_of(noad)) then
      point_at_family(s,
        math_family_for(s.char, family, settings_of(noad)))
    end
  elseif s.id == SUB_MLIST then
    rewrite_math_list(s.head, on, family)
  end
end

local function rewrite_noad (noad, enabled, family)
  if disabled(noad) then
    return
  end
  local on = enabled or settings_of(noad).cjk_math
  if not on or not family then
    return
  end
```

```
  rewrite_field(noad.nucleus, noad, on, family)
  for _, field in ipairs({ "num", "denom", "sup", "sub" }) do
    rewrite_field(noad[field], noad, on, family)
  end
end

rewrite_math_list = function (head, enabled, family)
  local cur = head
  while cur do
    rewrite_noad(cur, enabled, family)
    cur = cur.next
  end
  return head
end
```

Whether the one-time warning has been shown.

LuaCJK.warned_cjk_math

```
M.warned_cjk_math = M.warned_cjk_math or false
```

Warn once when a CJK character appears in math mode while the feature is off.

warn_disabled_cjk_math

Search a math list, including its sub lists, for a CJK character.

```
local function find_cjk_math (head)
```

find_cjk_math

```
  local cur = head
  while cur do
    local nuc = cur.nucleus
    if nuc then
      if (nuc.id == MATH_CHAR or nuc.id == MATH_TEXT_CHAR)
         and is_cjk(nuc.char, settings_of(cur)) then
        return true
      elseif nuc.id == SUB_MLIST and find_cjk_math(nuc.head) then
        return true
      end
    end
    for _, field in ipairs({ "num", "denom", "sup", "sub" }) do
      local sub = cur[field]
      if sub and sub.id == SUB_MLIST and find_cjk_math(sub.head) then
        return true
      end
    end
    cur = cur.next
  end
  return false
end
```

```
local function warn_disabled_cjk_math (head)
  if M.warned_cjk_math then
    return
  end
  if find_cjk_math(head) then
    M.warned_cjk_math = true
    texio.write_nl("term and log",
      "Package LuaCJK Warning: CJK character in math mode, "
      .. "but CJKmath is not enabled.")
    texio.write_nl("term and log",
      "(LuaCJK)                Use \\LuaCJKsetup{CJKmath = true} " ..
      "to typeset it with the CJK font.")
  end
end

local function math_family_of (head)
  local cur = head
  while cur do
    if M.attr_math_font then
      local family = has_attribute(cur, M.attr_math_font)
      if family and family > 0 then
        return family
      end
      local nuc = cur.nucleus
      if nuc then
        family = has_attribute(nuc, M.attr_math_font)
        if family and family > 0 then
          return family
        end
      end
    end
    cur = cur.next
  end
  return nil
end

local function rewrite_math (head)
  if not any_cjk_math_enabled(head) then
    warn_disabled_cjk_math(head)
    return head
  end
  return rewrite_math_list(head, true, math_family_of(head))
end

```

(*End of definition for* `rewrite_math` *and others.*)

A line that ends at a closing or middle mark, or that starts with an opening mark, measures the mark's advance short of its ink: the blank half stands in the margin. A line that never counted that blank refuses the mark it could have held, and is set loose instead. The hang therefore has to be known while the line is broken, and by then only the node list is left, so it is expressed as a zero-size rule of negative width beside the break: a closing mark that ends a line is followed by such a rule and the break glue carries the same width, so the two cancel while the line is set and the rule alone shortens the line when the break is taken; a glue and a rule in the same order let an opening mark begin a line. A kern would not do, because a kern that follows a character is a break opportunity of its own and would take a kinsoku penalty, while TEX never discards a rule at a line break.

Character protrusion is no help either, because it does not move the ink at the end of a line: `\protrudechars` together with `\rpcode` makes `\rightmarginkern` report the amount while the glyph keeps its position, measured at 10pt with a right code of 1000 in a short line, an overfull line, and a line whose width is the natural width. Only `\lpcode` moves the line, which is what the visible effect of microtype's protrusion on a Latin line consists of.

`bound_width` The advance that a mark keeps inside the line when it ends one, or nothing when the option is off. Without it the hang is the blank the font carries, or half of the advance when the file has no outline to read.

```
local function bound_width (settings)
  local value = settings and settings.bound_width
  if value then
    return value
  end
  return nil
end

```

(*End of definition for* `bound_width`.)

The hang of a mark is the blank its glyph carries, so the ink of a line-final or line-initial mark stands on the measure instead of a half em inside it. xeCJK reads that blank from the font with the XETEX primitive `\XeTeXglyphbounds`; LuaTeX has no such query, so the outline is read from the font file, once per file and once per glyph.

Only the tables a measurement needs are parsed, and the bytes of a glyph are read when it is first asked for, so a sixteen-megabyte CJK face does not sit in

memory a second time. The box is the one the control points span, which is what XeTeX reports as well.

font_files One entry per font file, holding its directory, its character map, and the boxes already measured.

```
local font_files = {}

```

(*End of definition for* `font_files`.)

u16 The big-endian integers of the file format.

i16

u32

```
local function u16 (s, o) return s:byte(o + 1) * 256 + s:byte(o + 2) end

local function i16 (s, o)
  local v = u16(s, o)
  if v >= 32768 then v = v - 65536 end
  return v
end

local function u32 (s, o) return u16(s, o) * 65536 + u16(s, o + 2) end
```

(*End of definition for* `u16`, `i16`, *and* `u32`.)

font_file The parsed directory of one font file, or nothing when it cannot be read. The handle stays open, because one file serves every mark of the document.

```
local function font_file (filename)
  local f = font_files[filename]
  if f ~= nil then
    return f or nil
  end
  local fh = io.open(filename, "rb")
  if not fh then
    font_files[filename] = false
    return nil
  end
  local function read (off, len)
    fh:seek("set", off)
    return fh:read(len) or ""
  end
  local head = read(0, 12)
  if #head < 12 then
    font_files[filename] = false
    return nil
  end
  local count = u16(head, 4)
```

```
  local dir = read(12, count * 16)
  local tables = {}
  for i = 0, count - 1 do
    local rec = i * 16
    local tag = dir:sub(rec + 1, rec + 4)
    tables[tag] = { offset = u32(dir, rec + 8), length = u32(dir, rec + 12) }
  end
  local t = { fh = fh, read = read, tables = tables, boxes = {} }
  if tables.head then
    t.upem = u16(read(tables.head.offset + 18, 2), 0)
  end
  if not (t.upem and t.upem > 0) then
    fh:close()
    font_files[filename] = false
    return nil
  end
  font_files[filename] = t
  return t
end
```

(*End of definition for* `font_file`.)

`font_cmap` The character map, preferring a Unicode subtable: a format 12 table for the third platform, then a format 4 one, then whichever remains.

```
local function font_cmap (f)
  local off = f.tables.cmap and f.tables.cmap.offset
  if not off then return end
  local head = f.read(off, 4)
  local n = u16(head, 2)
  local recs = f.read(off + 4, n * 8)
  local choice
  for i = 0, n - 1 do
    local p = i * 8
    local pid, eid = u16(recs, p), u16(recs, p + 2)
    local sub = off + u32(recs, p + 4)
    local fmt = u16(f.read(sub, 2), 0)
    local rank
    if fmt == 12 and pid == 3 and eid == 10 then rank = 3
    elseif fmt == 12 then rank = 2
    elseif fmt == 4 and pid == 3 and eid == 1 then rank = 2
    elseif fmt == 4 then rank = 1 end
    if rank and (not choice or rank > choice.rank) then
      choice = { rank = rank, sub = sub, format = fmt }
    end
  end
```

```
  if not choice then return end
  if choice.format == 4 then
    local head4 = f.read(choice.sub, 14)
    local segX2 = u16(head4, 6)
    local seg = segX2 // 2
    f.cmap = {
      format = 4, data = f.read(choice.sub, u16(head4, 2)), segments = seg,
      end_of = 14, start_of = 14 + segX2 + 2,
      delta_of = 14 + segX2 + 2 + segX2,
      range_of = 14 + segX2 + 2 + segX2 + segX2,
    }
  else
    local head12 = f.read(choice.sub, 16)
    f.cmap = { format = 12, data = f.read(choice.sub, u32(head12, 4)) }
  end
end
```

(*End of definition for* `font_cmap`.)

`font_glyph` The glyph index of a character, or zero.

```
local function font_glyph (f, code)
  if f.cmap then
    local c = f.cmap
    if c.format == 4 then
      local end_of, start_of, delta_of, range_of = c.end_of, c.start_of, c.delta_of, c
      for i = 0, c.segments - 1 do
        if code <= u16(c.data, end_of + i * 2) then
          local start = u16(c.data, start_of + i * 2)
          if code < start then return 0 end
          local delta = i16(c.data, delta_of + i * 2)
          local ro = u16(c.data, range_of + i * 2)
          if ro == 0 then return (code + delta) % 65536 end
          local addr = range_of + i * 2 + ro + (code - start) * 2
          local g = u16(c.data, addr)
          if g == 0 then return 0 end
          return (g + delta) % 65536
        end
      end
      return 0
    elseif c.format == 12 then
      local p = 16
      while p + 12 <= #c.data do
        local s, e, g = u32(c.data, p), u32(c.data, p + 4), u32(c.data, p + 8)
        if code >= s and code <= e then return g + (code - s) end
        p = p + 12
```

```
      end
      return 0
    end
    return 0
  end
  return 0
end
```

(*End of definition for* `font_glyph`.)

`font_advance` The advance of a glyph from `hmtx`; a glyph past the last entry takes the last advance, as the format says.

```
local function font_advance (f, gid)
  if f.advances then
    return f.advances[gid] or f.last_advance or 0
  end
  local hhea = f.tables.hhea and f.tables.hhea.offset
  local hmtx = f.tables.hmtx and f.tables.hmtx.offset
  if not (hhea and hmtx) then return f.upem end
  local nhm = u16(f.read(hhea + 34, 2), 0)
  if nhm == 0 then return f.upem end
  local data = f.read(hmtx, nhm * 4)
  local advances, last = {}, 0
  for i = 0, nhm - 1 do
    last = u16(data, i * 4)
    advances[i] = last
  end
  f.advances, f.last_advance = advances, last
  return advances[math.min(gid, nhm - 1)] or last
end
```

(*End of definition for* `font_advance`.)

`cff_index` A `CFF` index: where its data starts and the offset of each item, which is one-based from that start.

```
local function cff_index (f, p)
  local head = f.read(p, 3)
  local count = u16(head, 0)
  if count == 0 then return { count = 0, next = p + 2, data = p + 2 } end
  local off_size = head:byte(3)
  local offs = f.read(p + 3, (count + 1) * off_size)
  local function off (i)
    local v = 0
    local base = (i - 1) * off_size
    for k = 1, off_size do v = v * 256 + offs:byte(base + k) end
```

```
    return v
  end
  local data = p + 3 + (count + 1) * off_size
  return { count = count, data = data, off = off, next = data + off(count + 1) - 1 }
end
```

(*End of definition for* `cff_index`.)

`cff_string` One item of a CFF index, which is a charstring or a dictionary.

```
local function cff_string (f, idx, i)
  local a, b = idx.off(i), idx.off(i + 1)
  return f.read(idx.data + a - 1, b - a)
end
```

(*End of definition for* `cff_string`.)

`cff_dict` A CFF dictionary as its operators and operands. Only the two operators that point at the charstrings and the private dictionary are read, so a real number is skipped rather than converted.

```
local function cff_dict (s)
  local out, stack = {}, {}
  local i, n = 1, #s
  while i <= n do
    local b = s:byte(i)
    if b <= 21 then
      i = i + 1
      local op = b
      if b == 12 then
        op = 1200 + s:byte(i)
        i = i + 1
      end
      out[op] = stack
      stack = {}
    elseif b == 28 then
      local v = s:byte(i + 1) * 256 + s:byte(i + 2)
      if v >= 32768 then v = v - 65536 end
      stack[#stack + 1] = v
      i = i + 3
    elseif b == 29 then
      local v = s:byte(i + 1) * 16777216 + s:byte(i + 2) * 65536
                + s:byte(i + 3) * 256 + s:byte(i + 4)
      if v >= 2147483648 then v = v - 4294967296 end
      stack[#stack + 1] = v
      i = i + 5
    elseif b == 30 then
```

```
      i = i + 1
      while i <= n do
        local c = s:byte(i)
        i = i + 1
        if c % 16 == 15 then break end
      end
    elseif b >= 32 and b <= 246 then
      stack[#stack + 1] = b - 139
      i = i + 1
    elseif b >= 247 and b <= 250 then
      stack[#stack + 1] = (b - 247) * 256 + s:byte(i + 1) + 108
      i = i + 2
    elseif b >= 251 and b <= 254 then
      stack[#stack + 1] = -(b - 251) * 256 - s:byte(i + 1) - 108
      i = i + 2
    else
      i = i + 1
    end
  end
  return out
end
```

(*End of definition for* `cff_dict`.)

`cff_load` The charstrings of a `CFF` font and the two subroutine indexes a charstring may call into.

```
local function cff_load (f)
  local base = f.tables["CFF "] and f.tables["CFF "].offset
  if not base then return end
  local hdr = f.read(base + 2, 1)
  local p = base + hdr:byte(1)
  local names = cff_index(f, p)
  local tops = cff_index(f, names.next)
  local strings = cff_index(f, tops.next)
  local gsubrs = cff_index(f, strings.next)
  local top = cff_dict(cff_string(f, tops, 1))
  local cs = top[17] and top[17][1]
  if not cs then return end
  f.charstrings = cff_index(f, base + cs)
  f.gsubrs = gsubrs
  local priv = top[18]
  if priv then
    local size, offset = priv[1], priv[2]
    local pdict = cff_dict(f.read(base + offset, size))
    if pdict[19] then
```

```
      f.subrs = cff_index(f, base + offset + pdict[19][1])
    end
  end
  f.cff = true
end
```

(*End of definition for* `cff_load`.)

`cff_box` The horizontal extent of a Type 2 charstring. Every point of the outline is visited, the control points of a curve included, which is the box the font's own readers report; hints are counted only so that their masks can be skipped. A curve operator takes four operands, and the alternating ones start on the axis their name gives; the one operand of a $4n + 1$ run closes the last curve on the other axis. Each number is read from the charstring that holds it, since a subroutine carries its own operands, and the four bytes of the deprecated fixed form are read in full.

```
local function cff_box (f, cs)
  local x, xmin, xmax = 0, math.huge, -math.huge
  local stack, stems = {}, 0
  local function point (dx, dy)
    x = x + (dx or 0)
    if x < xmin then xmin = x end
    if x > xmax then xmax = x end
  end
  local function number (code, b, i)
    if b >= 32 and b <= 246 then return b - 139, i + 1 end
    if b >= 247 and b <= 250 then
      return (b - 247) * 256 + code:byte(i + 1) + 108, i + 2
    end
    if b >= 251 and b <= 254 then
      return -(b - 251) * 256 - code:byte(i + 1) - 108, i + 2
    end
    if b == 28 then
      local v = code:byte(i + 1) * 256 + code:byte(i + 2)
      if v >= 32768 then v = v - 65536 end
      return v, i + 3
    end
    if b == 255 then
      local v = code:byte(i + 1) * 16777216 + code:byte(i + 2) * 65536
                + code:byte(i + 3) * 256 + code:byte(i + 4)
      if v >= 2147483648 then v = v - 4294967296 end
      return v / 65536, i + 5
    end
    return nil, i
  end
```

```
local run
run = function (code, depth)
  local i, n = 1, #code
  while i <= n do
    local b = code:byte(i)
    local v
    v, i = number(code, b, i)
    if v ~= nil then
      stack[#stack + 1] = v
    else
      i = i + 1
      if b == 21 then
        point(stack[1], stack[2])
        stack = {}
      elseif b == 22 then
        point(stack[1], 0)
        stack = {}
      elseif b == 4 then
        point(0, stack[1])
        stack = {}
      elseif b == 5 then
        for k = 1, #stack - 1, 2 do point(stack[k], stack[k + 1]) end
        stack = {}
      elseif b == 6 or b == 7 then
        local horiz = (b == 6)
        for k = 1, #stack do
          if horiz then point(stack[k], 0) else point(0, stack[k]) end
          horiz = not horiz
        end
        stack = {}
      elseif b == 8 then
        for k = 1, #stack - 5, 6 do
          point(stack[k], stack[k + 1])
          point(stack[k + 2], stack[k + 3])
          point(stack[k + 4], stack[k + 5])
        end
        stack = {}
      elseif b == 24 then
        local k = 1
        while k <= #stack - 2 do
          point(stack[k], stack[k + 1])
          k = k + 2
        end
        point(stack[k], stack[k + 1])
        stack = {}
```

```
      elseif b == 25 then
        local k = 1
        while k <= #stack - 6 do
          point(stack[k], stack[k + 1])
          point(stack[k + 2], stack[k + 3])
          point(stack[k + 4], stack[k + 5])
          k = k + 6
        end
        point(stack[k], stack[k + 1])
        stack = {}
      elseif b == 26 or b == 27 then
        local k, first = 1, 0
        if #stack % 4 == 1 then first, k = stack[1], 2 end
        while k <= #stack - 3 do
          if b == 26 then
            point(first, stack[k])
            point(stack[k + 1], stack[k + 2])
            point(0, stack[k + 3])
          else
            point(stack[k], first)
            point(stack[k + 1], stack[k + 2])
            point(stack[k + 3], 0)
          end
          first = 0
          k = k + 4
        end
        stack = {}
      elseif b == 30 or b == 31 then
        local horiz = (b == 31)
        local k = 1
        while k <= #stack - 3 do
          if horiz then
            point(stack[k], 0)
            point(stack[k + 1], stack[k + 2])
            point(0, stack[k + 3])
          else
            point(0, stack[k])
            point(stack[k + 1], stack[k + 2])
            point(stack[k + 3], 0)
          end
          horiz = not horiz
          k = k + 4
        end
        if k == #stack then
          if horiz then
```

```
              point(stack[k], 0)
            else
              point(0, stack[k])
            end
          end
          stack = {}
        elseif b == 1 or b == 3 or b == 18 or b == 23 then
          stems = stems + #stack // 2
          stack = {}
        elseif b == 19 or b == 20 then
          stems = stems + #stack // 2
          stack = {}
          i = i + (stems + 7) // 8
        elseif b == 10 or b == 29 then
          local sub = table.remove(stack)
          local arr = (b == 10) and f.subrs or f.gsubrs
          if arr and arr.count > 0 and sub then
            local bias = (arr.count < 1240) and 107
                      or (arr.count < 33900 and 1131 or 32768)
            local idx = sub + bias + 1
            if idx >= 1 and idx <= arr.count and depth < 8 then
              run(cff_string(f, arr, idx), depth + 1)
            end
          end
        elseif b == 11 or b == 14 then
          return
        else
          stack = {}
        end
      end
    end
  end
  run(cs, 0)
  if xmin == math.huge then return nil end
  return xmin, xmax
end
```

(*End of definition for* `cff_box`.)

`font_box` The box of one character in font units: a `CFF` outline from its charstring, a `glyf` outline from the ten-byte header that starts it, and nothing for a character the font does not have.

```
local function font_box (f, code)
  local known = f.boxes[code]
  if known ~= nil then
```

```
    return known or nil
  end
  if f.cff == nil then cff_load(f) end
  if f.cmap == nil then font_cmap(f) end
  local gid = font_glyph(f, code)
  if not gid or gid == 0 then
    f.boxes[code] = false
    return nil
  end
  local advance = font_advance(f, gid)
  local xmin, xmax
  if f.charstrings then
    if gid > f.charstrings.count then return nil end
    xmin, xmax = cff_box(f, cff_string(f, f.charstrings, gid + 1))
  else
    local glyf = f.tables.glyf and f.tables.glyf.offset
    local loca = f.tables.loca and f.tables.loca.offset
    local head = f.tables.head and f.tables.head.offset
    if not (glyf and loca and head) then return nil end
    local long = u16(f.read(head + 50, 2), 0) == 1
    local function loc (i)
      if long then return u32(f.read(loca + i * 4, 4), 0) end
      return u16(f.read(loca + i * 2, 2), 0) * 2
    end
    local a, b = loc(gid), loc(gid + 1)
    if b <= a then return nil end
    local hdr = f.read(glyf + a, 10)
    xmin, xmax = i16(hdr, 2), i16(hdr, 6)
  end
  if not xmin then return nil end
  local box = { xmin = xmin, xmax = xmax, advance = advance }
  f.boxes[code] = box
  return box
end
```

(*End of definition for* `font_box`.)

`mark_blank` The blank of a mark on the side that hangs, in scaled points, or nothing when the file cannot be read or is one member of a collection. The advance is the one the style gives the mark, so a compressed mark is measured inside its own advance, where `\apply_fixed_widths` puts its ink.

```
local function mark_blank (g, mark, target)
  local font = font.getfont(g.font)
  if not (font and font.filename and font.size and (font.index or 0) == 0) then
    return nil
```

```
  end
  local f = font_file(font.filename)
  if not f then
    return nil
  end
  local box = font_box(f, g.char)
  if not box then
    return nil
  end
  local scale = font.size / f.upem
  local delta = target - box.advance * scale
  local blank
  if mark == "open" then
    blank = box.xmin * scale + delta
  else
    blank = target - box.xmax * scale
    if mark == "middle" then
      blank = blank - delta / 2
    end
  end
  if blank < 0 then
    blank = 0
  elseif blank > target then
    blank = target
  end
  return math.floor(blank + 0.5)
end
```

(*End of definition for* `mark_blank`.)

`hang_amount` How much of a mark's advance stands outside the line: the part that `PunctBoundWidth` leaves out, the blank the font carries for the mark, or half of the advance the style gives it when the file has no outline to read. The advance is the one `\apply_fixed_widths` gives it, so a compressed style hangs less.

```
local function hang_amount (g, mark, settings)
  local target = fixed_width(g, settings, style_of(settings)) or g.width
  local bound  = bound_width(settings)
  if bound then
    return math.max(0, target - bound)
  end
  local blank = mark_blank(g, mark, target)
  if blank then
    return blank
  end
```

```
  return math.floor(0.5 * target + 0.5)
end
```

(*End of definition for* `hang_amount`.)

`open_hang` `item_label` `head_lead` `hang_rule`

The hang of an opening mark that a line begins with, as a length and the glyph it belongs to, or zero when the mark does not open or its style does not hang; the rule that carries that length into the line, of negative width and with the attributes of the mark; and the head of a paragraph, found by walking over what carries no text of the paragraph: the local paragraph node, the whatsits, the indentation, which LuaTEX puts in front of the text as an empty box of the width of `\parindent`, and the label box of a list item, which the list writes in front of the item's text. A box is passed over when it carries no content, when it occupies no advance, or when a penalty of zero follows it, and a penalty is passed over unless it forces a break. The penalty of zero is the one the list writes after the label box of an item, so the label takes an advance in the line without being part of the paragraph, and a mark that begins the item hangs into it the way it hangs into the indentation of a paragraph.

```
local function open_hang (n)
  local lead, _, glyph = edge_mark(n, false)
  if not (glyph and not disabled(glyph) and lead == "open"
          and style_of(settings_of(glyph)).hang) then
    return 0, nil
  end
  return hang_amount(glyph, "open", settings_of(glyph)) or 0, glyph
end

local function item_label (b)
  local n = b.next
  return n ~= nil and n.id == PENALTY and n.penalty == 0
end

local function head_lead (head)
  local indent = tex.parindent or 0
  local prev   = nil
  local cur    = head
  while cur do
    if cur.id == GLYPH then
      return cur, prev
    elseif cur.id == HLIST or cur.id == VLIST then
      if cur.head and cur.width ~= 0 and not item_label(cur) then
        return nil, nil
      end
```

```
    elseif cur.id == GLUE then
      if cur.width ~= indent or cur.stretch ~= 0 or cur.shrink ~= 0 then
        return nil, nil
      end
    elseif cur.id == KERN then
      if cur.kern ~= 0 then
        return nil, nil
      end
    elseif cur.id == PENALTY then
      if cur.penalty < 0 then
        return nil, nil
      end
    elseif cur.id == MATH or cur.id == DISC or cur.id == RULE then
      return nil, nil
    end
    prev = cur
    cur  = cur.next
  end
  return nil, nil
end

local function hang_rule (width, glyph, prev, next_)
  local r  = new_node("rule")
  r.width  = -width
  r.height = 0
  r.depth  = 0
  inherit_attributes(r, glyph)
  r.next = next_
  if prev then
    prev.next = r
  end
  return r
end

```

(*End of definition for* `open_hang` *and others.*)

`hang_bounds` Bracket every break point that a mark ends or starts, so the line measures the hang and the text inside a line does not move. A break that a preceding penalty already forbids is left alone, so a fixed width does not carry a rule that can never be used. The two boundaries that carry no glue of their own are bracketed as well: the start of the paragraph, and the penalty that holds a break the document forced, where the rule that shifts the line is all there is to insert. Those two are left out when `\leftskip` reaches the line with a fill, which is how a centered or a flush-right paragraph is set: the line then begins at the centering position rather

than at the edge of the measure, and the rule would move it instead of leaving the mark's blank in the margin. The hang at a break stays in every case, because the line breaker has already counted it when it decided that the mark fits.

```
local function finite_skip (s)
  return not s or (s.stretch_order == 0 and s.shrink_order == 0)
end

local function hang_bounds (head)
  local left  = finite_skip(tex.leftskip)
  local first, before = head_lead(head)
  if first and left then
    local width, glyph = open_hang(first)
    if width > 0 then
      local r = hang_rule(width, glyph, before, first)
      if not before then
        head = r
      end
    end
  end
  local prev = nil
  local last = nil
  local cur  = head
  while cur do
    if cur.id == GLYPH
       or ((cur.id == HLIST or cur.id == VLIST) and box_visible(cur)) then
      last = cur
    elseif cur.id == PENALTY and cur.penalty <= -10000 then
      local nxt = next_visible(cur.next)
      local width, glyph = 0, nil
      if nxt and left then
        width, glyph = open_hang(nxt)
      end
      if width > 0 then
        hang_rule(width, glyph, cur, cur.next)
        cur = cur.next
      end
    elseif is_break_point(cur) and last and cur.id == GLUE
           and not (prev and prev.id == PENALTY and prev.penalty >= 10000) then
      local nxt                       = next_visible(cur.next)
      local tail, _, tail_glyph       = edge_mark(last, true)
      local w_end                     = 0
      local w_start, lead_glyph       = 0, nil
      if nxt then
        w_start, lead_glyph = open_hang(nxt)
```

```
      end
      if tail_glyph and not disabled(tail_glyph)
         and (tail == "close" or tail == "middle")
         and style_of(settings_of(tail_glyph)).hang then
        w_end = hang_amount(tail_glyph, tail, settings_of(tail_glyph))
      end
      if w_end > 0 or w_start > 0 then
        local delta = w_end + w_start
        cur.width = cur.width + delta
        if w_end > 0 then
          local r  = new_node("rule")
          r.width  = -w_end
          r.height = 0
          r.depth  = 0
          inherit_attributes(r, tail_glyph)
          r.next = cur
          if prev then
            prev.next = r
          else
            head = r
          end
          prev = r
        end
        if w_start > 0 then
          local r  = new_node("rule")
          r.width  = -w_start
          r.height = 0
          r.depth  = 0
          inherit_attributes(r, lead_glyph)
          r.next = cur.next
          cur.next = r
        end
      end
    end
    prev = cur
    cur  = cur.next
  end
  return head
end

```

(*End of definition for* `hang_bounds`.)

TEX can break at the glue that LuaCJK inserts between CJK characters. At the end of a paragraph the last two such break points get a penalty, so that at

least three CJK characters travel to the last line together instead of a single one.

apply_check_single Insert the widow penalties. Only a paragraph whose last visible node is a CJK glyph is examined, and only when its settings ask for it. The scan back to the previous character passes over what carries no text — a whatsit such as a color change, a mark, an insertion, the adjustment of a box, a penalty, and a kern this package inserted — so such a node does not end the run. The penalty goes in front of the glue that the break is taken at, and it is left out when a penalty already stands in front of that glue, so a second run over the same list adds nothing.

```
local function apply_check_single (head)
  local trail = {}
  local node  = head
  while node do
    trail[#trail + 1] = node
    node = node.next
  end
  local last
  for i = #trail, 1, -1 do
    local item = trail[i]
    if item.id == GLYPH then
      if is_cjk(item.char, settings_of(item), item.font) then
        last = i
      end
      break
    elseif item.id == HLIST or item.id == VLIST then
      break
    end
  end
  if not last then
    return head
  end
  local settings = settings_of(trail[last])
  if not settings.check_single then
    return head
  end
  local penalty  = settings.widow_penalty
  local inserted = 0
  local i        = last
  while i > 1 and inserted < 2 do
    local j = i - 1
    while j >= 1 and (trail[j].id == GLUE or transparent_node(trail[j])
                      or (trail[j].id == KERN and is_inserted(trail[j]))) do
      j = j - 1
    end
```

```
    if j < 1 then
      break
    end
    local glyph = trail[j]
    if glyph.id ~= GLYPH or not is_cjk(glyph.char, settings_of(glyph), glyph.font) the
      break
    end
    local at    = nil
    local taken = false
    for k = j + 1, i - 1 do
      local item = trail[k]
      if item.id == GLUE then
        at = k
        break
      elseif item.id == PENALTY then
        taken = true
      end
    end
    if at and not taken then
      local p = new_node("penalty")
      inherit_attributes(p, glyph)
      p.penalty          = penalty
      p.next             = trail[at]
      trail[at - 1].next = p
    end
    inserted = inserted + 1
    i        = j
  end
  return head
end

```

(*End of definition for* `apply_check_single`.)

`LuaCJK.process` Run the passes over a list, in the order in which they depend on each other: what shaping left of the zero-width characters first, so that the spacing pass sees the text as it is set; then the spacing, the fixed widths, the prohibitions, and the widow penalties. Without `pre_shaping_filter` the font is moved here instead, after shaping, because there was no pass in front of it. The hang of a mark is inserted only when the list is a paragraph about to be broken, because a box packed by `\hpack` has no break to hang at.

```
local function process (head, linebreak)
  if M.shaping then
    head = drop_orphans(head)
```

```
  else
    head = rewrite_fonts(head)
  end
  head = spacing(head)
  head = apply_fixed_widths(head)
  if linebreak then
    head = hang_bounds(head)
  end
  head = apply_kinsoku(head)
  head = apply_check_single(head)
  return head
end

M.process = process

```

(*End of definition for* `LuaCJK.process`.)

`LuaCJK.pre_linebreak` The line-breaking callback.

```
function M.pre_linebreak (head, groupcode)
  return process(head, true)
end

```

(*End of definition for* `LuaCJK.pre_linebreak`.)

`LuaCJK.hpack` The box-packing callback.

```
function M.hpack (head, groupcode, size, packtype, direction, attributelist)
  return process(head)
end

```

(*End of definition for* `LuaCJK.hpack`.)

`LuaCJK.pre_mlist_to_hlist` The math callback.

```
function M.pre_mlist_to_hlist (head, display, penalties)
  return rewrite_math(head)
end

```

(*End of definition for* `LuaCJK.pre_mlist_to_hlist`.)

`hyphenate_runs` Hyphenate the runs of a word that lie between its CJK characters. TEX counts a CJK character as a letter, so the patterns see a CJK character and the Latin word beside it as one word, and a point they find there puts a hyphen right next to the character. A CJK character ends a word here instead: the run of text on each side

of it is hyphenated on its own, so no point the patterns find can stand between a CJK character and the letters that follow, and the boundary keeps the glue that `CJKecglue` gives it.

```
local function hyphenate_runs (head, tail)
  local run  = nil
  local prev = nil
  local cur  = head
  while cur do
    if cur.id == GLYPH and is_cjk(cur.char, settings_of(cur), cur.font) then
      if run then
        lang.hyphenate(run, prev)
      end
      run = nil
    elseif not run then
      run = cur
    end
    prev = cur
    if cur == tail then
      break
    end
    cur = cur.next
  end
  if run then
    lang.hyphenate(run, prev)
  end
end

```

(*End of definition for* `hyphenate_runs`.)

`LuaCJK.hyphenate` The hyphenation callback. TEX hyphenates a word while it breaks the lines, after the pre-linebreaking pass has already built the list, so the discretionaries of a word appear after the prohibitions have been placed and nothing forbids them yet. The callback hyphenates exactly as TEX does, a run of non-CJK text at a time, and then reads the discretionaries it has just created, one word at a time.

```
function M.hyphenate (head, tail)
  hyphenate_runs(head, tail)
  local left = nil
  local cur  = head
  while cur do
    if cur.id == GLYPH then
      left = cur
    elseif cur.id == DISC then
      forbid_marked_disc(cur, left)
```

```
    end
    if cur == tail then
      break
    end
    cur = cur.next
  end
end
```

(*End of definition for* `LuaCJK.hyphenate`.)

The callbacks are registered in this order: the font substitution, which luaotfload calls from its own handler before it shapes a list; the spacing, which runs on both lists; the math; and the hyphenation, which is the only one that runs inside the line breaker rather than around it. The substitution is registered only when luaotfload has the callback; without it, `process` applies the font after shaping instead.

```
if M.shaping then
  luatexbase.add_to_callback("pre_shaping_filter",
    M.pre_shaping, "LuaCJK.font")
end
luatexbase.add_to_callback("pre_linebreak_filter",
  M.pre_linebreak, "LuaCJK.spacing")
luatexbase.add_to_callback("hpack_filter", M.hpack, "LuaCJK.spacing")
luatexbase.add_to_callback("pre_mlist_to_hlist_filter",
  M.pre_mlist_to_hlist, "LuaCJK.math")
luatexbase.add_to_callback("hyphenate", M.hyphenate, "LuaCJK.hyphenate")
⟨/lua-core⟩
```

## A.9 Decorations

lua-ul draws each decoration as a pair of nodes: a leaders glue of width $w$ and a glue of width $-w$ that gives the space back, so the text does not move. Changing those numbers is enough to inset the decoration and to leave out the punctuation, which is why LuaCJK adjusts the box instead of drawing the decoration itself. TEX numbers the glue subtypes, and 100 is `\leaders`, 101 `\cleaders`, and 102 `\xleaders`; lua-ul uses the first and this package adds the third where a grid cell may be spread, so the pass takes all three.

The emphasis dots are the exception: a pattern repeated along the span cannot follow the characters when the inter-character glue stretches, and it would dot Latin text as well, so the ink of `\CJKunderdot` goes into the text itself. Its leaders pair carries an empty box and keeps the width, the stretch, and the shrink of the

span, while the pass writes one dot behind each CJK character, centred on the advance of that character.

LuaCJKfntef.attr
LuaCJKfntef.settings
LuaCJKfntef.dot_attr
LuaCJKfntef.color_attr

The attribute that carries the number of a settings entry, and the entry itself: the inset at both ends, whether punctuation is skipped, and the geometry of the emphasis dots. A separate entry per call keeps two decorations in one paragraph apart. The dots carry an attribute of their own, so that a box the pass sees twice is not dotted twice, and the number of luacolor's attribute, which is what colors a rule.

```
4047 ⟨*lua-fntef⟩
4048 LuaCJKfntef = LuaCJKfntef or {}
4049
4050 local M        = LuaCJKfntef
4051 local GLYPH    = node.id("glyph")
4052 local GLUE     = node.id("glue")
4053 local LEADERS  = 100
4054 local CLEADERS = 101
4055 local XLEADERS = 102
4056 local KERN     = node.id("kern")
4057 local HLIST    = node.id("hlist")
4058 local VLIST    = node.id("vlist")
4059 local RULE     = node.id("rule")
4060 local WHATSIT  = node.id("whatsit")
4061
4062 M.attr       = M.attr     or nil
4063 M.settings   = M.settings or {}
4064 M.next_id    = M.next_id  or 1
4065 M.dot_attr   = M.dot_attr or luatexbase.new_attribute("LuaCJKfntef.dot")
4066 M.color_attr = luatexbase.attributes
4067   and luatexbase.attributes["LuaCol@Attribute"]
4068
```

(*End of definition for* `LuaCJKfntef.attr` *and others.*)

LuaCJKfntef.em_descent

The descent of the em box of the current font, in scaled points. A CJK em box is one em square, so its descent is both what a highlight has to add below the baseline to cover a character, and what separates the ink from the bottom of that square.

The value comes from `ascent`, `descent` and `units_per_em` of the font file, cached per file name: opening a CJK font costs about half a second, and one file serves every size.

```
4069 local em_metrics = {}
4070
```

```
local function font_em (filename)
  local m = em_metrics[filename]
  if m ~= nil then
    return m or nil
  end
  local ok, t = pcall(fontloader.open, filename)
  if ok and t and t.units_per_em and t.units_per_em > 0 and t.descent then
    m = { descent = t.descent, upem = t.units_per_em }
    fontloader.close(t)
  else
    m = false
  end
  em_metrics[filename] = m
  return m or nil
end

function M.em_descent (id)
  local f = tonumber(id)
  f = f and f > 0 and font.getfont(f) or nil
  local size = f and f.size or tex.sp("1em")
  local m = f and f.filename and font_em(f.filename)
  local sp
  if m then
    sp = math.floor(m.descent / m.upem * size + 0.5)
  else
    sp = math.floor(0.12 * size + 0.5)
  end
  return string.format("%d", sp)
end

```

(*End of definition for* `LuaCJKfntef.em_descent`.)

A decoration records its settings by selecting an entry: the number of the entry goes into the attribute on the decorated text, and attribute zero means that the text carries no decoration of this package.

`LuaCJKfntef.new_settings` Register one entry and select it. The entry carries the dots of `\CJKunderdot` when the command gave no symbol: the side of one dot, the distance from the baseline to its top, and the color attribute of `format`, zero when the dots follow the color of their text.

```
function M.new_settings (subtract, skip, hidden, dots, size, top, color)
  local id       = M.next_id
  M.next_id      = id + 1
  if dots then
```

```
    dots = { size = size or 0, top = top or 0, color = color or 0 }
  else
    dots = nil
  end
  M.settings[id] = { subtract = subtract, skip = skip, hidden = hidden,
                     dots = dots }
  if M.attr then
    tex.setattribute(M.attr, id)
  end
end
```

(*End of definition for* `LuaCJKfntef.new_settings`.)

`kind` The punctuation group of a character, as the core module classifies it.

```
local function kind (c)
  local class = LuaCJK.punct_class
  if class then
    return class(c)
  end
  return nil
end

```

(*End of definition for* `kind`.)

`node_width` The horizontal extent of a node, whether it stores it as a width or as a kern: a `kern` node has no `width` field in LuaTEX, and reading a nil where a dimension belongs put every later interval in the wrong place.

```
local function node_width (n)
  if n.id == KERN then
    return n.kern or 0
  end
  return n.width or 0
end

```

(*End of definition for* `node_width`.)

`segments` The intervals of a span of width ⟨*w*⟩ that the decoration may cover: the first and the last ⟨*inset*⟩ are taken off, and with ⟨*skip-punct*⟩ every punctuation mark inside is cut out as well. A cut is clamped to the span, so a mark that straddles the end keeps the part that fits.

```
local function segments (w, start, inset, skip_punct)
  local cuts = {}
  if skip_punct then
```

```
    local x = 0
    local g = start
    while g and x < w do
      local width = node_width(g)
      if g.id == GLYPH and kind(g.char) then
        cuts[#cuts + 1] = { x, math.min(x + width, w) }
      end
      x = x + width
      g = g.next
    end
  end
  local result = {}
  local low    = math.min(inset, w)
  local high   = math.max(w - inset, low)
  for _, cut in ipairs(cuts) do
    if cut[1] > low then
      result[#result + 1] = { low, math.min(cut[1], high) }
    end
    low = math.max(low, cut[2])
  end
  if low < high then
    result[#result + 1] = { low, high }
  end
  return result
end

```

(*End of definition for* `segments`.)

`settings_at` The settings of the decoration whose leaders pair ends just before ⟨*node*⟩: the number is read from that node, because the pair itself is inserted before the marked text.

```
local function settings_at (node_)
  if not node_ or not M.attr then
    return nil
  end
  local id = node.has_attribute(node_, M.attr)
  if not id or id == 0 then
    return nil
  end
  return M.settings[id]
end

```

(*End of definition for* `settings_at`.)

`single_rule` The rule that a decoration box reduces to, and how far the box moves it down. A box holding nothing but kerns, glues and one rule qualifies. `\leaders` repeats the box, and only whole copies fit, so the last fraction of the span stays empty: with a one-point object that is a visible gap at the right end of a highlight. A box that reduces to one rule is therefore drawn as a single rule of the full width instead.

```
local function single_rule (list, down)
  local found
  local n = list
  while n do
    if n.id == RULE then
      if found then
        return nil
      end
      found = n
    elseif n.id == HLIST then
      if found then
        return nil
      end
      local r, d = single_rule(n.head, down + (n.shift or 0))
      if not r then
        return nil
      end
      found, down = r, d
    elseif n.id == KERN then
      -- horizontal material, no vertical effect
    elseif n.id == GLUE and not n.leader then
      -- a plain glue between the pieces
    else
      return nil
    end
    n = n.next
  end
  if found then
    return found, down
  end
  return nil
end

local function box_rule (box)
  if not box or box.id ~= HLIST then
    return nil
  end
  local rule, down = single_rule(box.head, box.shift or 0)
```

```
  -- Only a solid block qualifies: the rule must fill the box, or the
  -- pattern would lose whatever sits beside it.
  if rule and rule.width == box.width then
    return rule, down
  end
  return nil
end

```

(*End of definition for* `single_rule`.)

`copy_attributes` Copy every attribute of one node onto another. The ink of a leaders box takes the attributes of the glue, not of the box it repeats, so a piece whose rule carries a color of its own would lose it.

```
local function copy_attributes (target, source)
  local a = source.attr
  while a do
    node.set_attribute(target, a.number, a.value)
    a = a.next
  end
end

```

(*End of definition for* `copy_attributes`.)

`hide_list` Take the ink out of the list a packed box holds. A glyph becomes a kern of the same width and a rule is dropped, because the box recorded its dimensions when it was packed and only the ink has to go; a box inside it is walked in turn, so the numerator of a fraction and the rule of its bar are hidden like a plain number. A box of the emphasis dots is left alone: its rule is the ink of the decoration, which `hidden` keeps, not of the text it covers.

```
local function hide_list (head)
  local prev, n = nil, head
  while n do
    local nxt = n.next
    if n.id == GLYPH then
      local k = node.new("kern")
      k.kern  = n.width or 0
      k.next  = nxt
      copy_attributes(k, n)
      if prev then
        prev.next = k
      else
        head = k
```

```
      end
      node.free(n)
      prev = k
    elseif n.id == RULE then
      if prev then
        prev.next = nxt
      else
        head = nxt
      end
      node.free(n)
    elseif n.id == HLIST
           and node.has_attribute(n, M.dot_attr) ~= nil then
      prev = n
    elseif n.id == HLIST or n.id == VLIST then
      n.head = hide_list(n.head)
      prev   = n
    else
      prev = n
    end
    n = nxt
  end
  return head
end

```

(*End of definition for* `hide_list`.)

`hide_text` Take the ink out of a decorated span and keep its dimensions. Every glyph becomes a kern of the same width, so the text still occupies the same space and the line breaks of the paragraph are the ones TeX found before the pass ran; one rule of no width carries the height and the depth, because a kern has neither. The span is the text the leaders pair covers, so the walk stops at the width of that pair. A box of the span has its own dimensions, so `hide_list` takes its ink out instead, and the box is left in the list where it stands.

```
local function hide_text (start, w)
  local x, height, depth = 0, 0, 0
  local prev = nil
  local n    = start
  while n and x < w do
    local nxt = n.next
    local width = node_width(n)
    if n.id == GLYPH then
      height = math.max(height, n.height or 0)
      depth  = math.max(depth,  n.depth  or 0)
```

```
      local k = node.new("kern")
      k.kern  = width
      copy_attributes(k, n)
      k.next  = nxt
      if prev then
        prev.next = k
      else
        start = k
      end
      node.free(n)
      prev = k
    else
      if n.id == HLIST or n.id == VLIST then
        height = math.max(height, n.height or 0)
        depth  = math.max(depth,  n.depth  or 0)
        if n.id == VLIST
           or node.has_attribute(n, M.dot_attr) == nil then
          n.head = hide_list(n.head)
        end
      end
      prev = n
    end
    x = x + width
    n = nxt
  end
  if height ~= 0 or depth ~= 0 then
    local r = node.new("rule")
    r.width, r.height, r.depth = 0, height, depth
    if prev then
      r.next    = prev.next
      prev.next = r
    else
      r.next = start
      start  = r
    end
  end
  return start
end
```

(*End of definition for* `hide_text`.)

`dot_box` One emphasis dot, written into the text after the character it belongs to. The box takes no space and adds neither height nor depth, so the decorated text keeps its width and the line keeps its leading; the dot is centred on the advance of the

character by a kern of half of it, and the inner box carries the ink at the depth of the call. The attributes travel from the character, so a dot takes the color of its text, and `format` replaces the color attribute with the value the TEX side resolved. The dot itself is marked, so that a box the pass sees twice is not dotted twice.

```
local function dot_box (glyph, dots)
  local outer = node.new("hlist")
  outer.width, outer.height, outer.depth = 0, 0, 0
  outer.dir = "TLT"
  local kern = node.new("kern")
  kern.kern  = -math.floor(((glyph.width or 0) + dots.size) / 2 + 0.5)
  local box = node.new("hlist")
  box.width, box.height, box.depth = dots.size, dots.size, 0
  box.shift = dots.top
  box.dir   = "TLT"
  local rule = node.new("rule")
  rule.width, rule.height, rule.depth = dots.size, dots.size, 0
  copy_attributes(kern, glyph)
  copy_attributes(box, glyph)
  copy_attributes(outer, glyph)
  copy_attributes(rule, glyph)
  if dots.color > 0 and M.color_attr then
    node.set_attribute(rule, M.color_attr, dots.color)
  end
  node.set_attribute(outer, M.dot_attr, 1)
  box.head   = rule
  kern.next  = box
  outer.head = kern
  return outer
end

```

(*End of definition for* `dot_box`.)

`insert_dots` Write one dot under each CJK character of the span of a leaders pair. The walk stops at the width of the pair, and an inner box of the text is walked with the interval of its own, so a character inside a box is dotted as well. A dot goes where the centre of its character lies inside the span that `subtract` leaves, and `skip` leaves the punctuation marks undotted; the two bounds are measured on the span and travel into a box unchanged, because the centre is measured from the start of the span and not from the start of the box. A character that already carries a dot behind it is passed over, because the pass also runs when a box is packed a second time.

```
local function insert_dots (start, w, settings, dots, offset, low, high)
```

```
  offset = offset or 0
  if not low then
    local subtract = settings.subtract or 0
    low  = math.min(subtract, w)
    high = math.max(w - subtract, low)
  end
  local x        = 0
  local n        = start
  while n and x < w do
    local nxt   = n.next
    local width = node_width(n)
    if n.id == GLYPH and width > 0
       and LuaCJK.is_cjk(n.char, LuaCJK.settings_of(n), n.font) then
      local centre = offset + x + width / 2
      local dotted = nxt and nxt.id == HLIST
                     and node.has_attribute(nxt, M.dot_attr) ~= nil
      if centre >= low and centre <= high and not dotted
         and not (settings.skip and kind(n.char)) then
        local dot = dot_box(n, dots)
        dot.next = nxt
        n.next   = dot
      end
    elseif n.id == HLIST then
      insert_dots(n.head, n.width or 0, settings, dots, offset + x,
        low, high)
    end
    x = x + width
    n = nxt
  end
end

```

(*End of definition for* `insert_dots`.)

`replace_pair` Replace the leaders glue of one decoration. The replacement covers the same span, with the same natural width, stretch, and shrink, in the same orders: the pieces are leaders of the intervals that `segments` allows, with their share of the dimensions of the pair, and kerns where a mark is cut out or the span is shortened. The last piece takes the remainder, so the sums are exact, and the negative glue that follows the leaders is kept as the tail.

A pair whose leader box reduces to one solid rule gets a leader box of zero width instead: TEX fills such a rule to the whole width of the glue, so the rule covers the interval exactly and grows with the glue, where a repeated rule box would leave

the last fraction of the interval empty.

```
local function replace_pair (head, prev, leaders, settings)
  local w     = leaders.width
  local neg   = leaders.next
  local subtract = settings and settings.subtract or 0
  local skip     = settings and settings.skip or false
  local parts = segments(w, neg.next, subtract, skip)
  if settings and settings.hidden then
    neg.next = hide_text(neg.next, w)
  end
  local rule, down = box_rule(leaders.leader)
  local block
  if rule then
    block = node.copy(rule)
    block.width  = 0
    block.height = rule.height - down
    block.depth  = rule.depth + down
    block.next   = nil
  end
  local covered = {}
  for _, part in ipairs(parts) do
    if part[2] > part[1] then
      covered[#covered + 1] = part
    end
  end
  local stretch = leaders.stretch or 0
  local shrink  = leaders.shrink or 0
  local given_st, given_sh = 0, 0
  local first, last
  local cursor = 0
  local function append (node_)
    if last then
      last.next = node_
    else
      first = node_
    end
    last = node_
  end
  local function gap (to)
    if to > cursor then
      local k = node.new("kern")
      k.kern  = to - cursor
      append(k)
      cursor = to
    end
```

```
  end
  for i, part in ipairs(covered) do
    gap(part[1])
    local piece    = node.copy(leaders)
    piece.width    = part[2] - part[1]
    piece.next     = nil
    if i < #covered then
      piece.stretch  = math.floor(stretch * piece.width / w + 0.5)
      piece.shrink   = math.floor(shrink  * piece.width / w + 0.5)
      given_st = given_st + piece.stretch
      given_sh = given_sh + piece.shrink
    else
      piece.stretch  = stretch - given_st
      piece.shrink   = shrink - given_sh
    end
    if block then
      piece.leader = node.copy(block)
      copy_attributes(piece, piece.leader)
    end
    append(piece)
    cursor = part[2]
  end
  gap(w)
  if #covered == 0 then
    -- Every interval was cut away, so the pair keeps its dimensions in one
    -- glue of no width, or the line would lose the stretch it had.
    local g = node.new("glue")
    g.width, g.stretch, g.shrink = 0, stretch, shrink
    g.stretch_order, g.shrink_order =
      leaders.stretch_order or 0, leaders.shrink_order or 0
    append(g)
  end
  if first then
    if prev then
      prev.next = first
    else
      head = first
    end
    last.next = neg
  end
  return head
end

```

(*End of definition for* `replace_pair`.)

adjust Walk a horizontal list and rewrite every leaders pair whose following node carries a settings entry. A pair of the emphasis dots keeps the nodes lua-ul made, because its leader box is empty and there is no pattern to place; what the pass adds is the ink, one dot behind each CJK character of the span.

```
local function adjust (head)
  if not M.attr then
    return head
  end
  local prev = nil
  local n    = head
  while n do
    local nxt = n.next
    if n.id == GLUE
       and (n.subtype == LEADERS or n.subtype == CLEADERS
            or n.subtype == XLEADERS)
       and n.next
       and n.next.id == GLUE and n.next.width == -n.width then
      local neg      = n.next
      local settings = settings_at(neg.next)
      if settings and settings.dots then
        insert_dots(neg.next, n.width, settings, settings.dots, 0)
        if settings.hidden then
          neg.next = hide_text(neg.next, n.width)
        end
        n    = neg
        nxt  = neg.next
      elseif settings or box_rule(n.leader) then
        head = replace_pair(head, prev, n, settings)
        n    = neg
        nxt  = neg.next
      end
    end
    prev = n
    n    = nxt
  end
  return head
end

```

(*End of definition for* `adjust`.)

glue_width The width a glue node actually has in a packed box. The node keeps its natural width, and the box records how much of the glue's stretch or shrink was used: `glue_set` is the fraction, and `glue_sign` says whether it is applied to the stretch

(1) or to the shrink (2), against the glue whose order is `glue_order`. This is the only place where the final width of a decoration is available, since it is fixed when the line is packed.

```
local function glue_width (box, g)
  local w    = g.width or 0
  local set  = box.glue_set or 0
  local sign = box.glue_sign or 0
  local order = box.glue_order or 0
  if sign == 1 and (g.stretch_order or 0) == order then
    w = w + math.floor(set * (g.stretch or 0) + 0.5)
  elseif sign == 2 and (g.shrink_order or 0) == order then
    w = w - math.floor(set * (g.shrink or 0) + 0.5)
  end
  return w
end

```

(*End of definition for* `glue_width`.)

`drawn` Whether a leader box holds a drawing rather than ink of its own. A drawing reaches the page as PDF literals, so the box holds whatsits and no glyph and no rule; the dots and the symbols hold a rule or a glyph, and a symbol wrapped in an `ActualText` sequence holds both.

```
local function drawn (box)
  if not box or box.id ~= HLIST then
    return false
  end
  local literal, ink = false, false
  local n = box.head
  while n do
    if n.id == GLYPH or n.id == RULE then
      ink = true
    elseif n.id == WHATSIT then
      literal = true
    elseif n.id == HLIST then
      local l, i = drawn(n)
      literal = literal or l
      ink     = ink or i
    end
    n = n.next
  end
  return literal, ink
end

```

(*End of definition for* `drawn`.)

`literal` One PDF literal, placed at the current point: the mode `0` the backend wraps in a translation, which is the mode l3draw uses for its own operations.

```
local function literal (data)
  local n = node.new("whatsit", node.subtype("pdf_literal"))
  n.data  = data
  n.mode  = 0
  return n
end

```

(*End of definition for* `literal`.)

`scale_ink` Make the period of a copy equal to the width of its slot. A leader box is one period wide, and the slot the append callback divides the piece into is almost never the same width: the two differ by the remainder divided by the number of copies, which is a hundredth of a point or less, but it is still a phase shift between two copies and shows as a hairline gap or a thin doubling in the drawing. The ink of the copy is therefore painted inside a scope that scales it horizontally by ⟨*width*⟩ divided by the period: the path then ends exactly where the next copy's begins, and its period is the slot's. The stroke keeps its thickness, because only the horizontal axis is scaled; the pen's horizontal extent changes by the same fraction.

```
local function scale_ink (box, scale, advance)
  if not box or box.id ~= HLIST or not box.head then
    return
  end
  if math.abs(scale - 1) < 0.0000005 then
    return
  end
  local q = literal("q " .. string.format("%.6f", scale) .. " 0 0 1 0 0 cm")
  local z = literal("Q 1 0 0 1 " .. string.format("%.6f", advance) .. " 0 cm")
  q.next = box.head
  local last = q
  while last.next do
    last = last.next
  end
  last.next  = z
  box.head   = q
end

```

(*End of definition for* `scale_ink`.)

`tile` Write the copies of a drawn decoration into the piece it covers. The append callback is the one place where the width of the piece is final, so there it is divided into the nearest whole number of patterns and the piece is rebuilt from copies of the leader box, each as wide as that division: the copies are adjacent, the pattern reaches both ends of the span, and its period follows the stretch of the line. The width is rounded down, so that the remainder is a few scaled points at most and is left as a kern, and the ink of a copy is scaled to the width it is given, so that one copy's path ends exactly where the next one's begins: the period of the drawing is one em, and the width the piece is divided into is almost never the same.
The copies are written here rather than repeated by `\leaders`, because the copies of a leader glue are not laid out from its left edge: a span that is not a whole number of patterns loses one at one end, which is what a decoration showed whose span ended in the middle of a stretched line. `\xleaders` cannot spread the pattern either, because the glue it puts between two copies separates two butt caps.
A copy also loses the record of the drawing's origin, which l3draw puts in front of every drawing: it is a box of no width that holds nothing but whatsits, and a label that every copy records is multiply defined.

```
local function only_whatsits (head)
  local n = head
  while n do
    if n.id ~= WHATSIT then
      return false
    end
    n = n.next
  end
  return true
end

local function strip_record (head)
  local n = head
  while n do
    local nxt = n.next
    if n.id == HLIST then
      if (n.width or 0) == 0 and n.head and n.head.id == WHATSIT
         and only_whatsits(n.head) then
        head = node.remove(head, n)
      else
        n.head = strip_record(n.head)
      end
    end
    n = nxt
```

```
  end
  return head
end

local function tile (box)
  local prev = nil
  local n    = box.head
  while n do
    local nxt = n.next
    if n.id == GLUE and n.subtype == LEADERS and n.leader then
      local literal, ink = drawn(n.leader)
      if literal and not ink then
        local w      = glue_width(box, n)
        local period = n.leader.width
        if period > 0 and w >= period then
          local k     = math.floor(w / period + 0.5)
          local width = math.floor(w / k)
          local first, last
          for _ = 1, k do
            local copy = node.copy(n.leader)
            copy.head  = strip_record(node.copy_list(n.leader.head))
            copy.width = width
            scale_ink(copy, width / period, period / 65536 * 72 / 72.27)
            copy_attributes(copy, n)
            copy.next = nil
            if last then
              last.next = copy
            else
              first = copy
            end
            last = copy
          end
          local rest = w - k * width
          if rest > 0 then
            local tail = node.new("kern")
            tail.kern  = rest
            last.next  = tail
            last       = tail
          end
          last.next = nxt
          if prev then
            prev.next = first
          else
            box.head  = first
          end
```

```
          n = last
        end
      end
    end
    prev = n
    n    = nxt
  end
end
```

(*End of definition for* `tile`.)

`LuaCJKfntef.hpack` The box-packing callback. The pass also runs before a box is appended to a vertical list, because a box built earlier is not packed again; that is where a decoration is divided into whole patterns.

```
function M.hpack (head)
  return adjust(head)
end

luatexbase.add_to_callback("hpack_filter", M.hpack, "LuaCJKfntef.adjust")
luatexbase.add_to_callback("pre_append_to_vlist_filter",
  function (box)
    if box.id == node.id("hlist") then
      box.head = adjust(box.head)
      tile(box)
    end
    return box
  end, "LuaCJKfntef.adjust")
⟨/lua-fntef⟩
```

(*End of definition for* `LuaCJKfntef.hpack`.)

The *LuaCJKfntef package is a thin layer over lua-ul, which provides the underlining engine. It is generated from this file and may be loaded on its own; it loads LuaCJK first.

```
⟨*fntef⟩
\sys_if_engine_luatex:F
  {
    \msg_new:nnn { LuaCJKfntef } { engine }
      { LuaCJKfntef ~ requires ~ LuaTeX. }
    \msg_critical:nn { LuaCJKfntef } { engine }
    \tex_endinput:D
  }
\RequirePackage { LuaCJK, lua-ul, luacolor, l3draw }
\lua_load_module:n { LuaCJKfntef }
```

```
\newattribute \g__LuaCJK_fntef_settings_attr
\int_const:Nn \c__LuaCJK_fntef_settings_attr_int { \int_use:N \allocationnumber }
\g__LuaCJK_fntef_settings_attr = 0
\lua_now:e
  { LuaCJKfntef.attr = \int_use:N \c__LuaCJK_fntef_settings_attr_int }
```

\CJKunderline

Underline the argument. The optional argument takes `thickness`, `depth`, `textformat`, and any other key is passed on to lua-ul, which understands `top`, `bottom`, and `height`. The requested geometry, the format of the text, the color that an underline type is cached by, the color operations of a drawn pattern, and the keys that lua-ul receives. The emphasis dots add whether the ink is a dot per CJK character rather than a pattern, the side and the position of one dot, and the color attribute of the dots when `format` is given.

```
\dim_new:N \l__LuaCJK_fntef_depth_dim
\dim_new:N \l__LuaCJK_fntef_thickness_dim
\tl_new:N \l__LuaCJK_fntef_textformat_tl
\tl_new:N \l__LuaCJK_fntef_format_tl
\dim_new:N \l__LuaCJK_fntef_subtract_dim
\tl_new:N \l__LuaCJK_fntef_skip_tl
\tl_new:N \l__LuaCJK_fntef_skip_js_tl
\tl_new:N \l__LuaCJK_fntef_hidden_js_tl
\bool_new:N \l__LuaCJK_fntef_silent_bool
\bool_new:N \l__LuaCJK_fntef_hidden_bool
\bool_new:N \l__LuaCJK_fntef_top_given_bool
\bool_new:N \l__LuaCJK_fntef_bottom_given_bool
\dim_new:N \l__LuaCJK_fntef_rule_height_dim
\dim_new:N \l__LuaCJK_fntef_rule_depth_dim
\prop_new:N \g__LuaCJK_fntef_format_prop
\int_new:N \g__LuaCJK_fntef_format_int
\tl_new:N \l__LuaCJK_fntef_cache_key_tl
\tl_new:N \l__LuaCJK_fntef_cache_name_tl
\tl_new:N \l__LuaCJK_fntef_color_key_tl
\tl_new:N \l__LuaCJK_fntef_color_ops_tl
\tl_new:N \g__LuaCJK_fntef_color_ops_tl
\tl_new:N \l__LuaCJK_fntef_pass_tl
\dim_new:N \l__LuaCJK_fntef_top_dim
\dim_new:N \l__LuaCJK_fntef_bottom_dim
\dim_new:N \l__LuaCJK_fntef_gap_dim
\dim_new:N \l__LuaCJK_fntef_height_dim
\tl_new:N \l__LuaCJK_fntef_symbol_tl
\bool_new:N \l__LuaCJK_fntef_dots_bool
\tl_new:N \l__LuaCJK_fntef_dots_js_tl
\dim_new:N \l__LuaCJK_fntef_dots_size_dim
\dim_new:N \l__LuaCJK_fntef_dots_top_dim
```

```
\int_new:N \g__LuaCJK_fntef_dots_color_int
\dim_set:Nn \l__LuaCJK_fntef_depth_dim     { .2em }
\dim_set:Nn \l__LuaCJK_fntef_thickness_dim { .4pt }
\dim_set:Nn \l__LuaCJK_fntef_gap_dim       { 1.1pt }
\dim_set:Nn \l__LuaCJK_fntef_height_dim    { .3em }
```

(*End of definition for* `\CJKunderline` *and others. This function is documented on page 39.*)

Every decoration takes the same options, so one key set serves them all: the keys a command does not use are accepted and ignored, and a bare item that is not a key is a color name, which is `format`.

`\__LuaCJK_fntef_option_pair:nn` `\__LuaCJK_fntef_option_bare:n` `\__LuaCJK_fntef_options:n`

Parse the option list of a decoration and reset the defaults first. A pair goes to the key set as it is; a bare item is a key when it exists and a color name otherwise, so `[subtract]` keeps its meaning and `[red]` is `[format = red]`.

```
\cs_new_protected:Npn \__LuaCJK_fntef_option_pair:nn #1#2
  { \keys_set:nn { LuaCJKfntef / decoration } { #1 = {#2} } }
\cs_new_protected:Npn \__LuaCJK_fntef_option_bare:n #1
  {
    \keys_if_exist:nnTF { LuaCJKfntef / decoration } {#1}
      { \keys_set:nn { LuaCJKfntef / decoration } {#1} }
      { \keys_set:nn { LuaCJKfntef / decoration } { format = #1 } }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_options:n #1
  {
    \__LuaCJK_fntef_reset_decoration:
    \tl_if_novalue:nF {#1}
      {
        \keyval_parse:NNn
          \__LuaCJK_fntef_option_bare:n \__LuaCJK_fntef_option_pair:nn {#1}
      }
  }
```

(*End of definition for* `\__LuaCJK_fntef_option_pair:nn`, `\__LuaCJK_fntef_option_bare:n`, *and* `\__LuaCJK_fntef_options:n`.)

`\__LuaCJK_fntef_body:n`

The text of a decoration, with the format of the call applied. Every command goes through this, so `textformat` means the same thing everywhere; `hidden` is a setting of the decoration pass instead, because a hidden text has to keep its line breaks.

```
\cs_new_protected:Npn \__LuaCJK_fntef_body:n #1
  { \l__LuaCJK_fntef_textformat_tl #1 }
```

(*End of definition for* `\__LuaCJK_fntef_body:n`.)

__LuaCJK_fntef_push_settings: Hand the decoration settings to Lua, or clear the attribute when they are all at their defaults. The dots are settings of their own: when `\CJKunderdot` gives no `symbol`, the pass puts one dot under each CJK character rather than repeating a pattern, so the entry carries the side of a dot, the distance from the baseline to its top, and the color attribute of `format`.

```
\cs_new_protected:Npn \__LuaCJK_fntef_push_settings:
  {
    \bool_lazy_and:nnTF
      { \l__LuaCJK_fntef_dots_bool }
      { \tl_if_empty_p:N \l__LuaCJK_fntef_symbol_tl }
      {
        \tl_set:Nn \l__LuaCJK_fntef_dots_js_tl { true }
        \dim_set:Nn \l__LuaCJK_fntef_dots_size_dim { .06em }
        \dim_set:Nn \l__LuaCJK_fntef_dots_top_dim
          { \LuaCJKemdescent + \l__LuaCJK_fntef_depth_dim }
        \tl_if_empty:NF \l__LuaCJK_fntef_format_tl
          {
            \group_begin:
              \luaul_set_color:n { \l__LuaCJK_fntef_format_tl }
              \luaul_apply_color:
              \int_gset:Nn \g__LuaCJK_fntef_dots_color_int
                { \tex_the:D \use:c { LuaCol@Attribute } }
            \group_end:
          }
      }
      { \tl_set:Nn \l__LuaCJK_fntef_dots_js_tl { false } }
    \bool_lazy_all:nTF
      {
        { \dim_compare_p:nNn { \l__LuaCJK_fntef_subtract_dim } = { 0pt } }
        { \str_if_eq_p:Vn \l__LuaCJK_fntef_skip_tl { 0 } }
        { \bool_not_p:n { \l__LuaCJK_fntef_hidden_bool } }
        { \str_if_eq_p:Vn \l__LuaCJK_fntef_dots_js_tl { false } }
      }
      {
        \lua_now:e
          {
            tex.setattribute
              ( \int_use:N \c__LuaCJK_fntef_settings_attr_int, 0 )
          }
      }
      {
        \str_if_eq:VnTF \l__LuaCJK_fntef_skip_tl { 1 }
          { \tl_set:Nn \l__LuaCJK_fntef_skip_js_tl { true  } }
          { \tl_set:Nn \l__LuaCJK_fntef_skip_js_tl { false } }
```

```
        \bool_if:NTF \l__LuaCJK_fntef_hidden_bool
          { \tl_set:Nn \l__LuaCJK_fntef_hidden_js_tl { true  } }
          { \tl_set:Nn \l__LuaCJK_fntef_hidden_js_tl { false } }
        \lua_now:e
          {
            LuaCJKfntef.new_settings
              ( \dim_to_decimal_in_sp:n { \l__LuaCJK_fntef_subtract_dim } ,
                \l__LuaCJK_fntef_skip_js_tl ,
                \l__LuaCJK_fntef_hidden_js_tl ,
                \l__LuaCJK_fntef_dots_js_tl ,
                \dim_to_decimal_in_sp:n { \l__LuaCJK_fntef_dots_size_dim } ,
                \dim_to_decimal_in_sp:n { \l__LuaCJK_fntef_dots_top_dim } ,
                \int_use:N \g__LuaCJK_fntef_dots_color_int )
          }
      }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_prepare_underline:n #1
  {
    \__LuaCJK_fntef_options:n {#1}
    \bool_if:NF \l__LuaCJK_fntef_top_given_bool
      { \dim_set:Nn \l__LuaCJK_fntef_top_dim { -\l__LuaCJK_fntef_depth_dim } }
    \bool_if:NF \l__LuaCJK_fntef_bottom_given_bool
      {
        \dim_set:Nn \l__LuaCJK_fntef_bottom_dim
          { -\l__LuaCJK_fntef_depth_dim - \l__LuaCJK_fntef_thickness_dim }
      }
    \tl_put_left:Nx \l__LuaCJK_fntef_pass_tl
      {
        top    = \dim_use:N \l__LuaCJK_fntef_top_dim ,
        bottom = \dim_use:N \l__LuaCJK_fntef_bottom_dim ,
      }
    \__LuaCJK_fntef_push_settings:
    \dim_set:Nn \l__LuaCJK_fntef_rule_height_dim
      { \l__LuaCJK_fntef_top_dim }
    \dim_set:Nn \l__LuaCJK_fntef_rule_depth_dim
      { -\l__LuaCJK_fntef_bottom_dim }
  }
```

A colored rule needs its own underline type: luacolor colors by attribute, while `\color` would emit whatsits in front of `\leaders`. The type is cached by color and geometry, so a document builds one type per distinct combination.

`\__LuaCJK_fntef_formatted_underline:`

```
\cs_new_protected:Npn \__LuaCJK_fntef_formatted_underline:
  {
    \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
      {
```

```
        \l__LuaCJK_fntef_format_tl
        | \dim_use:N \l__LuaCJK_fntef_rule_height_dim
        | \dim_use:N \l__LuaCJK_fntef_rule_depth_dim
      }
    \luaul_set_color:n { \l__LuaCJK_fntef_format_tl }
    \prop_get:NVNF \g__LuaCJK_fntef_format_prop
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_cache_name_tl
      {
        \int_gincr:N \g__LuaCJK_fntef_format_int
        \tl_set:Ne \l__LuaCJK_fntef_cache_name_tl
          { LuaCJKfntef / ul / \int_use:N \g__LuaCJK_fntef_format_int }
        \exp_args:Nc \newunderlinetype { \l__LuaCJK_fntef_cache_name_tl }
          [ \dim_to_decimal_in_sp:n { 1ex } @ \luaul_context_color: ]
          {
            \luaul_apply_color:
            \tex_leaders:D \tex_vrule:D
              height \l__LuaCJK_fntef_rule_height_dim
              depth \l__LuaCJK_fntef_rule_depth_dim
          }
        \prop_gput:NVV \g__LuaCJK_fntef_format_prop
          \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_cache_name_tl
      }
    \use:c { \l__LuaCJK_fntef_cache_name_tl }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_underLine:nn #1#2
  { \underLine [ #1 ] { #2 } }
\NewDocumentCommand \CJKunderline { s t- o +m }
  {
    \group_begin:
      \__LuaCJK_fntef_prepare_underline:n {#3}
      \tl_if_empty:NTF \l__LuaCJK_fntef_format_tl
        {
          \exp_args:NV \__LuaCJK_fntef_underLine:nn
            \l__LuaCJK_fntef_pass_tl { \__LuaCJK_fntef_body:n {#4} }
        }
        { \__LuaCJK_fntef_formatted_underline: { \__LuaCJK_fntef_body:n {#4} } }
    \group_end:
  }
```

(*End of definition for* `\__LuaCJK_fntef_push_settings:` *and* `\__LuaCJK_fntef_formatted_underline:`.)

`\CJKsout` Strike through the argument.

```
\cs_new_protected:Npn \__LuaCJK_fntef_use_sout:
  {
```

```
\tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
  {
    sout |
    \dim_use:N \l__LuaCJK_fntef_height_dim |
    \dim_use:N \l__LuaCJK_fntef_thickness_dim |
    \l__LuaCJK_fntef_format_tl
  }
\tl_if_empty:NF \l__LuaCJK_fntef_format_tl
  { \luaul_set_color:n { \l__LuaCJK_fntef_format_tl } }
\prop_get:NVNF \g__LuaCJK_fntef_format_prop
  \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_cache_name_tl
  {
    \int_gincr:N \g__LuaCJK_fntef_format_int
    \tl_set:Ne \l__LuaCJK_fntef_cache_name_tl
      { LuaCJKfntef / st / \int_use:N \g__LuaCJK_fntef_format_int }
    \tl_if_empty:NTF \l__LuaCJK_fntef_format_tl
      {
        \exp_args:Nc \newunderlinetype
          { \l__LuaCJK_fntef_cache_name_tl }
          {
            \tex_leaders:D \tex_vrule:D
              height \dim_eval:n
                {
                  \l__LuaCJK_fntef_height_dim
                  + \l__LuaCJK_fntef_thickness_dim / 2
                }
              depth \dim_eval:n
                {
                  -\l__LuaCJK_fntef_height_dim
                  + \l__LuaCJK_fntef_thickness_dim / 2
                }
          }
      }
      {
        \exp_args:Nc \newunderlinetype
          { \l__LuaCJK_fntef_cache_name_tl }
          [ \dim_to_decimal_in_sp:n { 1ex } @ \luaul_context_color: ]
          {
            \luaul_apply_color:
            \tex_leaders:D \tex_vrule:D
              height \dim_eval:n
                {
                  \l__LuaCJK_fntef_height_dim
                  + \l__LuaCJK_fntef_thickness_dim / 2
                }
```

```
                  depth \dim_eval:n
                    {
                      -\l__LuaCJK_fntef_height_dim
                      + \l__LuaCJK_fntef_thickness_dim / 2
                    }
                }
            }
          \prop_gput:NVV \g__LuaCJK_fntef_format_prop
            \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_cache_name_tl
        }
      \use:c { \l__LuaCJK_fntef_cache_name_tl }
    }
\NewDocumentCommand \CJKsout { s t- o +m }
  {
    \group_begin:
      \__LuaCJK_fntef_decoration_setup:n {#3}
      \__LuaCJK_fntef_use_sout:
      \__LuaCJK_fntef_body:n {#4}
    \group_end:
  }
```

(*End of definition for* `\CJKsout`*. This function is documented on page 42.*)

`__LuaCJK_fntef_use_double:` Build the two underline types of the double rule, one for each line, keyed by the geometry so that repeated calls reuse them.

```
\cs_new_protected:Npn \__LuaCJK_fntef_use_double:
  {
    \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
      {
        double-top |
        \dim_use:N \l__LuaCJK_fntef_depth_dim |
        \dim_use:N \l__LuaCJK_fntef_thickness_dim
      }
    \tl_set:Ne \l__LuaCJK_fntef_any_code_tl
      {
        \exp_not:N \tex_leaders:D \exp_not:N \tex_vrule:D
          height \dim_eval:n { -\l__LuaCJK_fntef_depth_dim }
          depth \dim_eval:n
            { \l__LuaCJK_fntef_depth_dim + \l__LuaCJK_fntef_thickness_dim }
      }
    \exp_args:NV \__LuaCJK_fntef_use_leader_type:nn
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_any_code_tl
    \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
      {
        double-bottom |
```

```
        \dim_use:N \l__LuaCJK_fntef_depth_dim |
        \dim_use:N \l__LuaCJK_fntef_thickness_dim |
        \dim_use:N \l__LuaCJK_fntef_gap_dim
      }
    \tl_set:Ne \l__LuaCJK_fntef_any_code_tl
      {
        \exp_not:N \tex_leaders:D \exp_not:N \tex_vrule:D
          height \dim_eval:n
            {
              -\l__LuaCJK_fntef_depth_dim
              - \l__LuaCJK_fntef_thickness_dim
              - \l__LuaCJK_fntef_gap_dim
            }
          depth \dim_eval:n
            {
              \l__LuaCJK_fntef_depth_dim
              + 2 \l__LuaCJK_fntef_thickness_dim
              + \l__LuaCJK_fntef_gap_dim
            }
      }
    \exp_args:NV \__LuaCJK_fntef_use_leader_type:nn
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_any_code_tl
  }
```

(*End of definition for* `\__LuaCJK_fntef_use_double:.`)

`\__LuaCJK_fntef_decoration_setup:n` The settings that the decoration pass reads. They are pushed on every call, so a command without options starts from the defaults instead of inheriting the previous call.

```
\keys_define:nn { LuaCJKfntef / decoration }
  {
    subtract  .code:n             =
      {
        \str_if_eq:nnTF {#1} { true }
          { \dim_set:Nn \l__LuaCJK_fntef_subtract_dim { .5em } }
          {
            \str_if_eq:nnTF {#1} { false }
              { \dim_zero:N \l__LuaCJK_fntef_subtract_dim }
              { \dim_set:Nn \l__LuaCJK_fntef_subtract_dim {#1} }
          }
      },
    subtract  .default:n          = { true },
    skip      .code:n             =
      {
        \str_if_eq:nnTF {#1} { true }
```

```
        { \tl_set:Nn \l__LuaCJK_fntef_skip_tl { 1 } }
        { \tl_set:Nn \l__LuaCJK_fntef_skip_tl { 0 } }
    },
  skip      .default:n          = { true },
  hidden    .bool_set:N         = \l__LuaCJK_fntef_hidden_bool,
  hidden    .default:n          = { true },
  selectable.bool_set_inverse:N = \l__LuaCJK_fntef_silent_bool,
  selectable.default:n          = { true },
  depth     .dim_set:N          = \l__LuaCJK_fntef_depth_dim,
  depth     .value_required:n   = true,
  thickness .dim_set:N          = \l__LuaCJK_fntef_thickness_dim,
  thickness .value_required:n   = true,
  gap       .dim_set:N          = \l__LuaCJK_fntef_gap_dim,
  gap       .value_required:n   = true,
  height    .dim_set:N          = \l__LuaCJK_fntef_height_dim,
  height    .value_required:n   = true,
  top       .code:n             =
    {
      \dim_set:Nn \l__LuaCJK_fntef_top_dim {#1}
      \bool_set_true:N \l__LuaCJK_fntef_top_given_bool
    },
  top       .value_required:n   = true,
  bottom    .code:n             =
    {
      \dim_set:Nn \l__LuaCJK_fntef_bottom_dim {#1}
      \bool_set_true:N \l__LuaCJK_fntef_bottom_given_bool
    },
  bottom    .value_required:n   = true,
  format    .tl_set:N           = \l__LuaCJK_fntef_format_tl,
  color     .tl_set:N           = \l__LuaCJK_fntef_format_tl,
  color     .value_required:n   = true,
  symbol    .tl_set:N           = \l__LuaCJK_fntef_symbol_tl,
  sep       .dim_set:N          = \l__LuaCJK_fntef_symbol_sep_dim,
  sep       .value_required:n   = true,
  textformat.tl_set:N           = \l__LuaCJK_fntef_textformat_tl,
  unknown   .code:n             =
    {
      \tl_put_right:Nx \l__LuaCJK_fntef_pass_tl
        {
          \l_keys_key_str
          \tl_if_empty:NF \l_keys_value_tl
            { = \exp_not:V \l_keys_value_tl }
          ,
        }
    },
```

```
  }
\cs_new_protected:Npn \__LuaCJK_fntef_reset_decoration:
  {
    \dim_zero:N \l__LuaCJK_fntef_subtract_dim
    \tl_set:Nn \l__LuaCJK_fntef_skip_tl { 0 }
    \bool_set_true:N \l__LuaCJK_fntef_silent_bool
    \bool_set_false:N \l__LuaCJK_fntef_hidden_bool
    \bool_set_false:N \l__LuaCJK_fntef_top_given_bool
    \bool_set_false:N \l__LuaCJK_fntef_bottom_given_bool
    \bool_set_false:N \l__LuaCJK_fntef_dots_bool
    \int_gzero:N \g__LuaCJK_fntef_dots_color_int
    \dim_set:Nn \l__LuaCJK_fntef_depth_dim     { .2em }
    \dim_set:Nn \l__LuaCJK_fntef_thickness_dim { .4pt }
    \dim_set:Nn \l__LuaCJK_fntef_gap_dim       { 1.1pt }
    \dim_set:Nn \l__LuaCJK_fntef_height_dim    { .3em }
    \clist_map_inline:nn { format, symbol, textformat, pass }
      { \tl_clear:c { l__LuaCJK_fntef_##1_tl } }
    \dim_set:Nn \l__LuaCJK_fntef_symbol_sep_dim { .1em }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_decoration_setup:n #1
  {
    \__LuaCJK_fntef_options:n {#1}
    \__LuaCJK_fntef_push_settings:
  }
```

(*End of definition for* `\__LuaCJK_fntef_decoration_setup:n`.)

`_LuaCJK_fntef_dots_setup:n` The same for the emphasis dots, which are a setting of their own: the options are parsed first, because the reset clears the flag, and the flag is set before the settings go to Lua, because the entry carries the side, the position, and the color of a dot.

```
\cs_new_protected:Npn \__LuaCJK_fntef_dots_setup:n #1
  {
    \__LuaCJK_fntef_options:n {#1}
    \bool_set_true:N \l__LuaCJK_fntef_dots_bool
    \__LuaCJK_fntef_push_settings:
  }
```

(*End of definition for* `\__LuaCJK_fntef_dots_setup:n`.)

`\CJKunderdblline` Two parallel rules. The optional argument takes the whole key set of the other decorations; `*` and `-` are accepted for compatibility.

```
\NewDocumentCommand \CJKunderdblline { s t- o +m }
  {
    \group_begin:
      \__LuaCJK_fntef_decoration_setup:n {#3}
```

```
      \__LuaCJK_fntef_use_double:
      \__LuaCJK_fntef_body:n {#4}
    \group_end:
  }
```

(*End of definition for* `\CJKunderdblline`. *This function is documented on page 40.*)

`\CJKunderwave` A wave drawn with l3draw, one period per em. The drawing is kept as code rather than as a box, so the `em` units are resolved when the decoration is used and the pattern follows the current font size.

The drawing goes through `\__LuaCJK_fntef_use_drawn_type:nn` rather than `\__LuaCJK_fntef_use_leader_type:nn`, because its ink is stroked and not a rule that luacolor could color by attribute; the color travels into the drawing itself, so the pattern is built once per font size and color.

The pattern is a pair of arches, each of them the quadratic that runs from one zero crossing to the next one through its apex, written as the cubic that is the same curve: its control points sit at a third and two thirds of the half period, two thirds of the amplitude high, so the curve leaves and meets the zero line at the same slope on both sides of a crossing and the wave has no cusp. The amplitude is a quarter em, so the wave is half an em high from crest to trough, half as high as its period is wide, which is the proportion the reference implementation gives it.

The period is forced to exactly one em by `\hbox to 1em`. l3draw sizes its box by the bounding box of the stroked path, which is a full stroke width wider than the path; repeating that box would leave a gap of one stroke width between two periods. The two `\hss` absorb the excess symmetrically, so no overfull box is reported. The stroke is set to `.04em`, which is `0.4pt` at 10pt, so the pattern keeps its look from the previous release while it scales with the font size.

The period is the width of a CJK character, so a span of whole characters that justification has not stretched, which is the last line of a paragraph, is an exact number of periods: its wave starts at the first character's edge, ends at the last one's, and puts a crest at the same place in every character.

A span that justification has stretched is another matter: TEX repeats only whole copies of a leader box, so the remainder of such a span is the stretch the piece carries, which at one em is up to a character. `\leaders` would leave all of it empty at the right end and `\cleaders` half of it at each end, where the wave stops before the last character. `\xleaders` divides it between the copies, but that glue separates two butt caps: a cap ends where its path ends, so every gap is a hole in the stroke, and at a tenth of a point it is already visible at the crossings. The piece is therefore divided into whole patterns instead, by `tile` (see the decoration pass): it has the final width and the nearest whole number of periods, and it writes one copy of the leader box per period at that width. The wave then reaches the last

character, no glue falls between two periods, and the period follows the line: on a paragraph set ten centimetres wide it is `9.92pt` on a line that was shrunk by a per cent and `10pt` on the last line, which was not stretched at all.
The width the piece is divided into is not the period of the drawing, so the ink of the copy is painted inside a scope that scales it horizontally to that width. Without it the drawing keeps its own period and two copies are a hundredth of a point out of phase, which is a hairline gap or a thin doubling at every crossing; with it one copy's path ends exactly where the next one's begins. The glue the piece is made of has no shrink of its own, so its final width is read from the line's own `glue_set` and `glue_sign`, the latter saying whether the line was stretched or shrunk. A box that never reaches a vertical list, such as `\hbox to`, keeps the plain `\leaders` behaviour: it stops after its last whole pattern, without breaking the stroke. A scratch box for the drawn patterns. Each pattern is built here and then has its height and depth set to zero: the ink is a PDF literal, so only the reported dimensions change, and a decoration never opens up the line it sits on.

`\l__LuaCJK_fntef_pattern_box`

```
\box_new:N \l__LuaCJK_fntef_pattern_box
\dim_new:N \l__LuaCJK_fntef_symbol_kern_dim
\dim_new:N \l__LuaCJK_fntef_symbol_raise_dim
\dim_new:N \l__LuaCJK_fntef_em_descent_dim

\cs_new_protected:Npn \__LuaCJK_fntef_wave:
  {
    \tl_set:Nn \l__LuaCJK_fntef_any_code_tl
      {
        \tex_leaders:D \hbox_to_wd:nn { 1em }
          {
            \tex_hss:D
            \hbox_set:Nn \l__LuaCJK_fntef_pattern_box
              {
                \box_move_up:nn { -.42em }
                  {
                    \hbox:n
                      {
                        \draw_begin:
                          \__LuaCJK_fntef_pattern_color:
                          \draw_set_linewidth:n { .04em }
                          \draw_path_moveto:n { 0em , 0em }
                          \draw_path_curveto:nnn { .1667em , .1667em }
                            { .3333em , .1667em } { .5em , 0em }
                          \draw_path_curveto:nnn { .6667em , -
.1667em }
                            { .8333em , -.1667em } { 1em , 0em }
                          \draw_path_use_clear:n { stroke }
```

```
                        \draw_end:
                      }
                  }
              }
            \box_set_ht:Nn  \l__LuaCJK_fntef_pattern_box { \c_zero_dim }
            \box_set_dp:Nn  \l__LuaCJK_fntef_pattern_box { \c_zero_dim }
            \box_use_drop:N \l__LuaCJK_fntef_pattern_box
            \tex_hss:D
          }
      }
    \tl_set:Nn \l__LuaCJK_fntef_cache_key_tl { wave }
    \exp_args:NV \__LuaCJK_fntef_use_drawn_type:nn
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_any_code_tl
  }
\cs_new_protected:Npn \__LuaCJK_fntef_silent_begin:
  {
    \bool_if:NT \l__LuaCJK_fntef_silent_bool
      {
        \tex_pdfextension:D literal direct
          { /Span <</ActualText <FEFF200B>>> BDC }
      }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_silent_end:
  {
    \bool_if:NT \l__LuaCJK_fntef_silent_bool
      { \tex_pdfextension:D literal direct { EMC } }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_use_symbol:n #1
  {
    \dim_set:Nn \l__LuaCJK_fntef_any_depth_dim {#1}
    \dim_set:Nn \l__LuaCJK_fntef_em_descent_dim
      {
        \lua_now:e
          {
            tex.print(LuaCJKfntef.em_descent
              ( \int_use:N \tex_attribute:D \c__LuaCJK_font_attr_int ))
          }
        sp
      }
    \tl_set:Ne \l__LuaCJK_fntef_any_code_tl
      {
        \exp_not:N \tex_leaders:D \exp_not:N \hbox:n
          {
            \exp_not:N \hbox_set:Nn \exp_not:N \l__LuaCJK_fntef_pattern_box
              { \exp_not:V \l__LuaCJK_fntef_symbol_tl }
```

```
            \exp_not:N \dim_set:Nn \exp_not:N \l__LuaCJK_fntef_symbol_kern_dim
              {
                \exp_not:N \dim_eval:n
                  {
                    ( 1em
                      - \exp_not:N \box_wd:N \l__LuaCJK_fntef_pattern_box ) / 2
                  }
              }
            \exp_not:N \dim_set:Nn \exp_not:N \l__LuaCJK_fntef_symbol_raise_dim
              {
                \exp_not:N \dim_eval:n
                  {
                    - \dim_use:N \l__LuaCJK_fntef_em_descent_dim
                    - \exp_not:N \l__LuaCJK_fntef_any_depth_dim
                    - \exp_not:N \box_ht:N \l__LuaCJK_fntef_pattern_box
                  }
              }
            \exp_not:N \box_set_ht:Nn \exp_not:N \l__LuaCJK_fntef_pattern_box
              { \exp_not:N \c_zero_dim }
            \exp_not:N \box_set_dp:Nn \exp_not:N \l__LuaCJK_fntef_pattern_box
              { \exp_not:N \c_zero_dim }
            \exp_not:N \skip_horizontal:n
              { \exp_not:N \l__LuaCJK_fntef_symbol_kern_dim }
            \exp_not:N \__LuaCJK_fntef_silent_begin:
            \exp_not:N \box_move_up:nn
              { \exp_not:N \l__LuaCJK_fntef_symbol_raise_dim }
              { \exp_not:N \box_use_drop:N \l__LuaCJK_fntef_pattern_box }
            \exp_not:N \__LuaCJK_fntef_silent_end:
            \exp_not:N \skip_horizontal:n
              { \exp_not:N \l__LuaCJK_fntef_symbol_kern_dim }
          }
      }
    \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
      {
        symbol |
        \dim_use:N \l__LuaCJK_fntef_em_descent_dim |
        \dim_use:N \l__LuaCJK_fntef_any_depth_dim |
        \bool_if:NTF \l__LuaCJK_fntef_silent_bool { silent } { text } |
        \tl_to_str:V \l__LuaCJK_fntef_symbol_tl
      }
    \exp_args:NV \__LuaCJK_fntef_use_leader_type:nn
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_any_code_tl
  }
\NewDocumentCommand \CJKunderwave { s t- o +m }
  {
```

```
    \group_begin:
      \__LuaCJK_fntef_decoration_setup:n {#3}
      \tl_if_empty:NTF \l__LuaCJK_fntef_symbol_tl
        { \__LuaCJK_fntef_wave: } { \__LuaCJK_fntef_use_symbol:n { .42em } }
      \__LuaCJK_fntef_body:n {#4}
    \group_end:
  }
```

(*End of definition for* `\CJKunderwave` *and* `\l__LuaCJK_fntef_pattern_box`*. This function is documented on page 41.*)

`\CJKxout` A repeated slash, as in ulem's `\xout`. The drawing is kept as code so that it scales with the font size, and the stroke width is `.04em` like the wave. The slash runs from `-.15em` to `.8em`, so it crosses the whole body of a CJK glyph instead of stopping in its upper half.

l3draw moves a drawing so that the bottom of its bounding box sits on the baseline of the box, which shifts a path that goes below the baseline upwards. The `\raise` of `-.17em` takes the slash back down by the part that lies below the baseline (`.15em`) plus half the stroke, so the ink lands where the path says.

Like the wave, the slash goes through `\__LuaCJK_fntef_use_drawn_type:nn`, so it is stroked in the color of `format` or of its text, and it is divided into whole cells, each written by the pass, for the reason given there.

```
\cs_new_protected:Npn \__LuaCJK_fntef_slash:
  {
    \tl_set:Nn \l__LuaCJK_fntef_any_code_tl
      {
        \tex_leaders:D \hbox:n
          {
            \hbox_set:Nn \l__LuaCJK_fntef_pattern_box
              {
                \box_move_up:nn { -.17em }
                  {
                    \hbox:n
                      {
                        \draw_begin:
                          \__LuaCJK_fntef_pattern_color:
                          \draw_set_linewidth:n { .04em }
                          \draw_path_moveto:n { 0em , -.15em }
                          \draw_path_lineto:n { .42em , .8em }
                          \draw_path_use_clear:n { stroke }
                        \draw_end:
                      }
                  }
              }
```

```
            \box_set_ht:Nn  \l__LuaCJK_fntef_pattern_box { \c_zero_dim }
            \box_set_dp:Nn  \l__LuaCJK_fntef_pattern_box { \c_zero_dim }
            \box_use_drop:N \l__LuaCJK_fntef_pattern_box
          }
      }
    \tl_set:Nn \l__LuaCJK_fntef_cache_key_tl { xout }
    \exp_args:NV \__LuaCJK_fntef_use_drawn_type:nn
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_any_code_tl
  }
\NewDocumentCommand \CJKxout { s t- o +m }
  {
    \group_begin:
      \__LuaCJK_fntef_decoration_setup:n {#3}
      \__LuaCJK_fntef_slash:
      \__LuaCJK_fntef_body:n {#4}
    \group_end:
  }
```

(*End of definition for* `\CJKxout`. *This function is documented on page 41.*)

`\LuaCJKemdescent` The depth of the em box of the current font, as a dimension. A CJK em box is one em square, so an object of height `1em` placed at this depth covers a character the way a PDF reader selects it.

```
\cs_new:Npn \LuaCJKemdescent
  {
    \lua_now:e
      {
        tex.print(LuaCJKfntef.em_descent
          ( \int_use:N \tex_attribute:D \c__LuaCJK_font_attr_int ))
      }
    sp
  }
```

(*End of definition for* `\LuaCJKemdescent`. *This function is documented on page 41.*)

`\CJKunderanyline` Underline with an arbitrary object: the object is put in a box, lowered by ⟨*depth*⟩, and repeated by `\leaders`. The underline type is cached by the object, the depth, and the color, so repeating a command with the same arguments costs nothing. The leader code being built and its depth.

`__LuaCJK_fntef_any_code_tl`
`\l__LuaCJK_fntef_any_depth_dim`

```
\tl_new:N \l__LuaCJK_fntef_any_code_tl
\dim_new:N \l__LuaCJK_fntef_any_depth_dim

\NewDocumentCommand \CJKunderanyline { m m +m }
  {
    \group_begin:
```

```
      \__LuaCJK_fntef_decoration_setup:n { }
      \dim_set:Nn \l__LuaCJK_fntef_any_depth_dim { -#1 }
      \tl_set:Ne \l__LuaCJK_fntef_any_code_tl
        {
          \exp_not:N \tex_leaders:D \exp_not:N \hbox:n
            {
              \exp_not:N \box_move_up:nn
                { \dim_use:N \l__LuaCJK_fntef_any_depth_dim }
                { \exp_not:N \hbox:n { \exp_not:n {#2} } }
            }
        }
      \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
        { anyline | #1 | \tl_to_str:n {#2} }
      \exp_args:NV \__LuaCJK_fntef_use_leader_type:nn
        \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_any_code_tl
      \__LuaCJK_fntef_body:n {#3}
    \group_end:
  }
```

(*End of definition for* `\CJKunderanyline`*,* `\l__LuaCJK_fntef_any_code_tl`*, and* `\l__LuaCJK_fntef_any_depth_dim`*. This function is documented on page 41.*)

`\__LuaCJK_fntef_use_leader_type:nn`

Build the underline type of a decoration that is made of `\leaders`, which repeats one rule or one symbol along the decorated text, and use it. A type is built once per leader code and color and then reused, so a document pays for each combination of geometry and color only once.

The type keeps the color that was in force when it was built: luacolor colors a rule by attribute, and lua-ul puts the attributes of the built box on every leader it inserts. A type cached by the geometry alone would therefore keep the color of the first call, so the color is part of the key: `format` resolved to the operations of its color, which is what lua-ul puts in the context of its own type, and otherwise the value of luacolor's attribute, which is the color of the surrounding text. A requested color is also applied inside the box, so that it holds wherever the leader is placed. The key holds the resolved operations rather than the name of the color, because a name may be defined again: `\colorlet{mark}{red}` followed by `\colorlet{mark}{blue}` gives two keys, and the second decoration is drawn in blue instead of reusing the type that the first one built.

The name of the type already carries a counter, so the context that lua-ul puts in the name of the type it defines is a constant: one that expands again when the type is used would rebuild the type with whatever color happened to be current then.

A drawn pattern takes its color another way. Its ink comes from l3draw, which strokes with the color that l3color holds, so luacolor's attribute never reaches it;

an attribute on a leader box that holds a drawing only leaks, because luacolor writes the color in front of the repeated box and finds no node after it to restore the previous one. The color operations are therefore read from `\current@color`, the resolved command that luacolor itself would write, and the drawing emits them inside its own graphics scope. A `format` is resolved in a group of its own, so the attribute it sets ends with that group and no text outside the pattern takes its color.

```
\cs_new_protected:Npn \__LuaCJK_fntef_use_leader_type:nn #1#2
  { \__LuaCJK_fntef_use_type:nnn {#1} {#2} { attribute } }
\cs_new_protected:Npn \__LuaCJK_fntef_use_drawn_type:nn #1#2
  { \__LuaCJK_fntef_use_type:nnn {#1} {#2} { drawn } }
\cs_new_protected:Npn \__LuaCJK_fntef_use_type:nnn #1#2#3
  {
    \tl_if_empty:NTF \l__LuaCJK_fntef_format_tl
      {
        \tl_set:Ne \l__LuaCJK_fntef_color_key_tl
          { ambient | \tex_the:D \use:c { LuaCol@Attribute } }
      }
      {
        \luaul_set_color:n { \l__LuaCJK_fntef_format_tl }
        \tl_set:Ne \l__LuaCJK_fntef_color_key_tl
          { format | \luaul_context_color: }
      }
    \str_if_eq:nnT {#3} { drawn } { \__LuaCJK_fntef_capture_color: }
    \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
      {
        \exp_not:n {#1}
        | \dim_to_decimal_in_sp:n { 1em }
        | \l__LuaCJK_fntef_color_key_tl
      }
    \prop_get:NVNF \g__LuaCJK_fntef_format_prop
      \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_cache_name_tl
      {
        \int_gincr:N \g__LuaCJK_fntef_format_int
        \tl_set:Ne \l__LuaCJK_fntef_cache_name_tl
          { LuaCJKfntef / ul / \int_use:N \g__LuaCJK_fntef_format_int }
        \str_if_eq:nnTF {#3} { drawn }
          {
            \exp_args:Nc \newunderlinetype
              { \l__LuaCJK_fntef_cache_name_tl } [ LuaCJKfntef ] {#2}
          }
          {
            \tl_if_empty:NTF \l__LuaCJK_fntef_format_tl
              {
```

```
                \exp_args:Nc \newunderlinetype
                  { \l__LuaCJK_fntef_cache_name_tl } [ LuaCJKfntef ] {#2}
              }
              {
                \exp_args:Nc \newunderlinetype
                  { \l__LuaCJK_fntef_cache_name_tl } [ LuaCJKfntef ]
                  { \luaul_apply_color: #2 }
              }
          }
        \prop_gput:NVV \g__LuaCJK_fntef_format_prop
          \l__LuaCJK_fntef_cache_key_tl \l__LuaCJK_fntef_cache_name_tl
      }
    \use:c { \l__LuaCJK_fntef_cache_name_tl }
  }
```

(*End of definition for* `\__LuaCJK_fntef_use_leader_type:nn`.)

`\__LuaCJK_fntef_capture_color:`
`\__LuaCJK_fntef_pattern_color:`
`\__LuaCJK_fntef_literal_direct:n`

Read the color operations of the pattern. Without `format` they are the ones of the text the pattern decorates, which `\current@color` already holds; with it, the color is resolved the way luacolor resolves it and read from the same place, in a group that keeps the attribute to itself. The drawing then writes them, expanded, so that the literal keeps the color of the call that built it: a literal whose payload still held the variable would expand it at shipout, with whatever color is current then.

```
\cs_new_protected:Npn \__LuaCJK_fntef_capture_color:
  {
    \tl_if_empty:NTF \l__LuaCJK_fntef_format_tl
      { \tl_set_eq:Nc \l__LuaCJK_fntef_color_ops_tl { current@color } }
      {
        \group_begin:
          \luaul_apply_color:
          \tl_gset_eq:Nc \g__LuaCJK_fntef_color_ops_tl { current@color }
        \group_end:
        \tl_set_eq:NN \l__LuaCJK_fntef_color_ops_tl \g__LuaCJK_fntef_color_ops_tl
      }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_pattern_color:
  {
    \tl_if_empty:NF \l__LuaCJK_fntef_color_ops_tl
      {
        \exp_args:NV \__LuaCJK_fntef_literal_direct:n
          \l__LuaCJK_fntef_color_ops_tl
      }
  }
```

```
\cs_new_protected:Npn \__LuaCJK_fntef_literal_direct:n #1
  {
    \tex_expanded:D
      { \exp_not:N \tex_pdfextension:D literal direct {#1} }
  }
```

(*End of definition for* `\__LuaCJK_fntef_capture_color:`, `\__LuaCJK_fntef_pattern_color:`, *and* `\__LuaCJK_fntef_literal_direct:n`.)

`\CJKunderanysymbol` One symbol per em, which is what one wants for a repeated mark: the cell is as wide as a CJK character, so a row of symbols lines up with the characters above, and the symbol is centered in its cell. The depth is handed to `\LuaCJKfntef_use_symbol:n`, which puts the top of the symbol that far below the baseline.

```
\NewDocumentCommand \CJKunderanysymbol { O{} m m +m }
  {
    \group_begin:
      \__LuaCJK_fntef_decoration_setup:n {#1}
      \tl_set:Nn \l__LuaCJK_fntef_symbol_tl {#3}
      \__LuaCJK_fntef_use_symbol:n {#2}
      \__LuaCJK_fntef_body:n {#4}
    \group_end:
  }
```

(*End of definition for* `\CJKunderanysymbol`. *This function is documented on page* 42.)

`\CJKunderdot` Emphasise with dots below the glyphs. The optional argument takes the key set of `\CJKunderline`, like the other decorations. A symbol given with `symbol` is a pattern repeated along the span, as `\CJKunderanysymbol` repeats its own; the default dot is a setting of the decoration pass, which puts one under each CJK character, so the ink travels with the character rather than with the beginning of the span.

```
\NewDocumentCommand \CJKunderdot { s t- o +m }
  {
    \group_begin:
      \__LuaCJK_fntef_dots_setup:n {#3}
      \tl_if_empty:NTF \l__LuaCJK_fntef_symbol_tl
        { \__LuaCJK_fntef_underdot:n { \__LuaCJK_fntef_body:n {#4} } }
        {
          \__LuaCJK_fntef_use_symbol:n { .3em }
          \__LuaCJK_fntef_body:n {#4}
        }
    \group_end:
  }
```

__LuaCJK_fntef_underdot:n

Apply the dots inside a group. The underline type of the dots carries no ink: it gives the decoration the pair of nodes that lua-ul provides, and with them the width, the stretch, and the shrink of the text. The pass then writes the dots into the text itself, which is also where `subtract` and `skip` act: a character whose centre lies in the part that `subtract` takes off gets no dot, and `skip` leaves the punctuation undotted. The underline type of the dots: a leader box of nothing, which leaves the pair without a pattern to repeat.

__LuaCJK_fntef_dots

```
\newunderlinetype \__LuaCJK_fntef_dots { \tex_leaders:D \hbox:n { } }

\cs_new_protected:Npn \__LuaCJK_fntef_underdot:n #1
  {
    \group_begin:
      \__LuaCJK_fntef_dots
      #1
    \group_end:
  }
```

(*End of definition for* `\CJKunderdot`, `\__LuaCJK_fntef_underdot:n`, *and* `\__LuaCJK_fntef_dots`. *This function is documented on page 42.*)

`CJKfilltwosides` (*env.*)

Set the body in a vertical box whose lines are justified to the given width, with the lines separated by `\\`. The inter-character glue is given an infinite stretch inside the box, so a line of CJK characters reaches the width even when there is no Latin word space to stretch. The body leaves vertical mode first, as `\parbox` does, so the box joins the paragraph that follows and the text after the environment continues on its line; a space written after the environment is ignored, as xeCJK ignores it. The content is therefore one marked-content sequence of that paragraph, and the automatic paragraph hooks are off around the box, so the box opens no paragraph of its own inside it. The environment takes `\\` back from an enclosing alignment, as `\parbox` does with `\@normalcr`, so a cell of a table can hold it: the break belongs to the box, and the alignment does not see it. The optional argument places the box on the baseline like the same argument of `\parbox`: `t` for the first line, `c` for the text axis (the default), and `b` for the last line. A width of zero or less keeps the surrounding line width. A `t` box is rebuilt as a `\vtop`, so that its reference point is the baseline of its first line, and a `b` box keeps its natural height and depth, whose reference point is the baseline of the last line. A `c` box is centred on the axis of the text style, `\Umathaxis`, the axis `\parbox` centres on; `\check@mathfonts` runs first, as it does there, because the axis reads the largest dimension until the math fonts exist. A `\vtop` takes the height of the first item of its list, so the marked content opens inside the first line rather than in the list in front of it; in front of it, a `t` box would have no height at all.

\l__LuaCJK_fntef_fill_box
\l__LuaCJK_fntef_fill_width_dim
\l__LuaCJK_fntef_fill_total_dim
\l__LuaCJK_fntef_fill_axis_dim
\l__LuaCJK_fntef_fill_position_tl

The box being filled, its requested width, its total height and depth, the axis of the text style, and the requested baseline position.

```
\box_new:N \l__LuaCJK_fntef_fill_box
\dim_new:N \l__LuaCJK_fntef_fill_width_dim
\dim_new:N \l__LuaCJK_fntef_fill_total_dim
\dim_new:N \l__LuaCJK_fntef_fill_axis_dim
\tl_new:N \l__LuaCJK_fntef_fill_position_tl
```

(*End of definition for* `\l__LuaCJK_fntef_fill_box` *and others.*)

```
\cs_new_protected:Npn \__LuaCJK_fntef_fill_begin:nn #1#2
  {
    \mode_leave_vertical:
    \group_begin:
    \cs_set_eq:Nc \\ { @normalcr }
    \cs_if_exist:NT \tagpdfparaOff { \tagpdfparaOff }
    \tl_set:Nn \l__LuaCJK_fntef_fill_position_tl {#1}
    \dim_set:Nn \l__LuaCJK_fntef_fill_width_dim {#2}
    \LuaCJKsetup { CJKglue = 0pt plus 1filll }
    \vbox_set:Nw \l__LuaCJK_fntef_fill_box
      \dim_compare:nNnT { \l__LuaCJK_fntef_fill_width_dim } > { 0pt }
        { \dim_set:Nn \tex_hsize:D { \l__LuaCJK_fntef_fill_width_dim } }
      \tex_noindent:D
      \cs_if_exist:NT \tag_mc_begin:n { \tag_mc_begin:n { } }
      \tex_ignorespaces:D
  }
\cs_new_protected:Npn \__LuaCJK_fntef_fill_end:
  {
    \tex_par:D
    \cs_if_exist:NT \tag_mc_end: { \tag_mc_end: }
    \vbox_set_end:
    \str_case:Vn \l__LuaCJK_fntef_fill_position_tl
      {
        { t }
        {
          \vbox_set_top:Nn \l__LuaCJK_fntef_fill_box
            { \vbox_unpack:N \l__LuaCJK_fntef_fill_box }
        }
        { c }
        {
          \use:c { check@mathfonts }
          \dim_set:Nn \l__LuaCJK_fntef_fill_total_dim
            { \box_ht_plus_dp:N \l__LuaCJK_fntef_fill_box }
          \dim_set:Nn \l__LuaCJK_fntef_fill_axis_dim
            { \tex_Umathaxis:D \textstyle }
          \box_set_ht:Nn \l__LuaCJK_fntef_fill_box
```

```
            { \l__LuaCJK_fntef_fill_total_dim / 2 + \l__LuaCJK_fntef_fill_axis_dim }
          \box_set_dp:Nn \l__LuaCJK_fntef_fill_box
            { \l__LuaCJK_fntef_fill_total_dim - \box_ht:N \l__LuaCJK_fntef_fill_box }
        }
      }
    \box_use_drop:N \l__LuaCJK_fntef_fill_box
    \cs_if_exist:NT \tagpdfparaOn { \tagpdfparaOn }
    \group_end:
    \ignorespacesafterend
  }
\NewDocumentEnvironment { CJKfilltwosides } { O{c} m }
  { \__LuaCJK_fntef_fill_begin:nn {#1} {#2} }
  { \__LuaCJK_fntef_fill_end:               }
```

`CJKfilltwosides*` (*env.*) The starred form behaves like the plain one; a width of zero or less keeps the surrounding line width in both.

```
\NewDocumentEnvironment { CJKfilltwosides* } { O{c} m }
  { \__LuaCJK_fntef_fill_begin:nn {#1} {#2} }
  { \__LuaCJK_fntef_fill_end:               }
⟨/fntef⟩
```

## A.10 Listings and ctex

The block listing environments are patched by making the CJK characters active while the character table of a listing is built. The active character is defined through the `\lccode` of two helper characters: the token in the body keeps category code 12, so it can be stored in the token list of listings and typeset later. The definitions are made once; each listing only sets the category codes again in its own group. A character outside the table is typeset the moment listings reads it, which puts it next to the wrong text and counts it as one column.

A run of CJK characters is one token. It is appended to the token list character by character, and the run is output only where listings ends a letter run: at a Latin letter, a digit, another character, or a space. The `emph` and `keywords` options match an entry against that token, so a CJK word is highlighted the way xeCJK highlights it. A CJK mark is a token of its own, because the word in front of it would otherwise enter the token with it; the marks are the ones LuaCJK.punct_codes lists inside the character table.

The two appending macros of listings are wrapped to end the run, and the saved macro has to be the last token of the wrapper: `\lst@Append` takes the character that follows the active character in the input as its argument, so a brace or the `\fi` of a conditional behind it would be taken instead.

The marks arrive from Lua as a comma-separated list of code points, printed with the catcode table of other characters: by the time the character table is built, listings has made every printable ASCII character active, so a digit printed with the current table would not be a number.

The ranges are those of `\LuaCJK.is_cjk`, limited to the Basic Multilingual Plane: each definition costs an entry of the string pool, and the supplementary planes would triple it for ideographs a listing almost never holds. A range includes its first and its last code point, since a mark that opens a range would otherwise return to the character handling above.

```
⟨*listings⟩
\RequirePackage { LuaCJK, listings }
\makeatletter
\newif\ifLuaCJK@listings@cjkrun
\newif\ifLuaCJK@listings@defined
\lst@AddToHook{InitVars}{\LuaCJK@listings@cjkrunfalse}
\def\LuaCJK@listings@process#1{%
  \lst@whitespacefalse
  \LuaCJK@listings@setcjkflag
  \advance\lst@length\@ne
  \lst@Append#1}
\def\LuaCJK@listings@processpunct#1{%
  \lst@whitespacefalse
  \LuaCJK@listings@cjkrunfalse
  \LuaCJK@listings@setcjkflag
  \advance\lst@length\@ne
  \lst@Append#1%
  \lst@Output
  \lst@letterfalse
  \LuaCJK@listings@cjkrunfalse}
\def\LuaCJK@listings@setcjkflag{%
  \ifLuaCJK@listings@cjkrun
    \lst@lettertrue
  \else
    \lst@ifletter \lst@Output \else \lst@OutputOther \lst@lettertrue \fi
    \LuaCJK@listings@cjkruntrue
  \fi}
\let\LuaCJK@listings@letter\lst@AppendLetter
\let\LuaCJK@listings@other\lst@AppendOther
\def\lst@AppendLetter{%
  \ifLuaCJK@listings@cjkrun
    \lst@Output \lst@lettertrue \LuaCJK@listings@cjkrunfalse
  \fi
  \LuaCJK@listings@letter}
```

```
\def\lst@AppendOther{%
  \ifLuaCJK@listings@cjkrun
    \lst@Output \lst@letterfalse \LuaCJK@listings@cjkrunfalse
  \fi
  \LuaCJK@listings@other}
\def\LuaCJK@listings@define#1#2{%
  \@tempcnta=#1\relax
  \loop
    \lccode`\~=\@tempcnta
    \lccode`\/=\@tempcnta
    \lowercase{\gdef~{\LuaCJK@listings@process/}}%
    \ifnum\@tempcnta<#2\relax
    \advance\@tempcnta\@ne
  \repeat}
\def\LuaCJK@listings@definepunct#1{%
  \lccode`\~=#1\relax
  \lccode`\/=#1\relax
  \lowercase{\gdef~{\LuaCJK@listings@processpunct/}}}
\def\LuaCJK@listings@catcodes#1#2{%
  \@tempcnta=#1\relax
  \loop
    \catcode\@tempcnta=13\relax
    \ifnum\@tempcnta<#2\relax
    \advance\@tempcnta\@ne
  \repeat}
\def\LuaCJK@listings@ranges#1{%
  #1{"00B7}{"00B7}%
  #1{"1100}{"11FF}%
  #1{"2014}{"2014}%
  #1{"2015}{"2015}%
  #1{"2018}{"2019}%
  #1{"201C}{"201D}%
  #1{"2025}{"2025}%
  #1{"2026}{"2026}%
  #1{"2027}{"2027}%
  #1{"22EF}{"22EF}%
  #1{"2E80}{"9FFF}%
  #1{"A960}{"A97C}%
  #1{"AC00}{"D7C6}%
  #1{"D7CB}{"D7FB}%
  #1{"F900}{"FAFF}%
  #1{"FE30}{"FE4F}%
  #1{"FF00}{"FFEF}}
\def\LuaCJK@listings@punctuate#1,{%
  \ifx\relax#1\relax\else
```

```
    \LuaCJK@listings@definepunct{#1}%
    \expandafter\LuaCJK@listings@punctuate
  \fi}
\def\LuaCJK@listings@punctuation{%
  \edef\LuaCJK@listings@codes{%
    \directlua{tex.sprint(-2,table.concat(LuaCJK.punct_codes(),","))}}%
  \expandafter\LuaCJK@listings@punctuate\LuaCJK@listings@codes,\relax,}
\def\LuaCJK@listings@activate{%
  \ifLuaCJK@listings@defined\else
    \LuaCJK@listings@ranges\LuaCJK@listings@define
    \global\LuaCJK@listings@definedtrue
  \fi
  \LuaCJK@listings@ranges\LuaCJK@listings@catcodes
  \LuaCJK@listings@punctuation}
```

`\lstinline` reads its argument with changed category codes, so the patch is limited to the block listing environments.

```
\lst@AddToHook{SelectCharTable}{%
  \ifx\lst@inlinechars\@empty
    \LuaCJK@listings@activate
  \fi}
\lst@AddToHook{OutputBox}{%
  \ifx\lst@inlinechars\@empty
    \LuaCJKdisable
  \fi}
\makeatother
⟨/listings⟩
```

ctex inputs an engine file named `ctex-engine-`⟨*name*⟩`.def`, where ⟨*name*⟩ is the detected engine unless the `engine` option names another. LuaCJK's file is therefore called `ctex-engine-luacjk.def` and is selected with `engine = luacjk`. The middle part must differ from `luatex`: ctex installs `ctex-engine-luatex.def`, a second file of that name would collide on CTAN, and the two files would describe different backends under one name.

The file defines the handful of `\ctex_`⟨*name*⟩ commands that the ctex kernel calls, and maps the parts of the luatexja interface that ctex and some documents still use.

```
⟨*ctex⟩
\typeout{LuaCJK~ctex~engine~loaded}
\sys_if_engine_luatex:F
  {
    \msg_new:nnn { LuaCJK } { ctex-engine }
      { The ~ ctex ~ engine ~ file ~ of ~ LuaCJK ~ requires ~ LuaTeX. }
    \msg_critical:nn { LuaCJK } { ctex-engine }
```

```
    \tex_endinput:D
  }
```

CJKfntef loads the legacy CJK package, which redefines the font family interface LuaCJK owns, so this file refuses it in both orders: the run stops when CJKfntef is already loaded, and the `package/CJKfntef/before` hook stops it when it is requested later. The message points at LuaCJKfntef, which does the same work through lua-ul. ctex's own LuaTeX backend refuses the package for the same reason, and that code does not run when the engine file is ours.

```
\msg_new:nnnn { LuaCJK } { cjkfntef }
  {
    Package ~ `CJKfntef' ~ is ~ not ~ compatible ~ with ~ the ~ LuaCJK ~
    backend ~ of ~ ctex.
  }
  { Use ~ the ~ `LuaCJKfntef' ~ package ~ instead. }
\cs_if_exist:cT { ver@CJKfntef.sty }
  { \msg_critical:nn { LuaCJK } { cjkfntef } }
\hook_gput_code:nnn { package / CJKfntef / before } { LuaCJK }
  { \msg_critical:nn { LuaCJK } { cjkfntef } }
```

The file loads the package it serves.

```
\RequirePackage { LuaCJK }
```

The fontset of ctex defines these shortcuts itself and checks for an existing definition, so the ones of LuaCJK have to go first.

```
\clist_map_inline:nn
  { \songti, \heiti, \fangsong, \kaishu, \lishu, \youyuan, \yahei, \pingfang }
  { \cs_undefine:N #1 }
```

⟨/ctex⟩

The interface that the ctex kernel and its classes rely on.

```
⟨*ctex⟩
\cs_new_protected:Npn \ctex_add_to_selectfont:n #1
  { \hook_gput_code:nnn { selectfont } { LuaCJK } {#1} }
\cs_new_eq:NN \ctex_ltj_zero_globaldefs: \prg_do_nothing:
\cs_new_protected:Npn \CTEX@alchar #1 {#1}
\cs_new_eq:NN \CTEX@beginallalchar \prg_do_nothing:
\cs_new_eq:NN \CTEX@endallalchar \prg_do_nothing:
\clist_map_inline:nn { \ccwd, \cht, \cdp, \cwd } { \dim_new:N #1 }
\cs_new_protected:Npn \ctex_update_ccwd:
  { \dim_set:Nn \ccwd { \f@size \p@ } }
\cs_new_protected:Npn \ctex_update_em_unit:
  { \dim_set:Nn \ccwd { \f@size \p@ } }
\cs_new_protected:Npn \ctex_update_kanjisize:
```

```
  {
    \dim_set:Nn \cwd { \ccwd }
    \dim_set:Nn \cht { \f@size \p@ }
    \dim_set:Nn \cdp { \c_zero_dim }
  }
⟨/ctex⟩
```

ctex keeps its own copy of the CJK glue so that it can tell whether the document changed it. The two copies are kept in step here.

```
⟨*ctex⟩
\skip_new:N \l__ctex_ccglue_skip
\skip_set_eq:NN \l__ctex_ccglue_skip \l__LuaCJK_ccglue_skip
\cs_new_protected:Npn \ctex_ltj_set_kanjiskip:N #1
  {
    \skip_set_eq:NN \l__LuaCJK_ccglue_skip #1
    \bool_set_true:N \l__LuaCJK_ccglue_touched_bool
    \__LuaCJK_settings_push:
  }
\cs_new_protected:Npn \ctex_ltj_set_xkanjiskip:N #1
  {
    \skip_set_eq:NN \l__LuaCJK_ecglue_skip #1
    \bool_set_true:N \l__LuaCJK_ecglue_touched_bool
    \__LuaCJK_settings_push:
  }
\cs_new_protected:Npn \ctex_update_ccglue:
  {
    \skip_set_eq:NN \l__LuaCJK_ccglue_skip \l__ctex_ccglue_skip
    \bool_set_true:N \l__LuaCJK_ccglue_touched_bool
    \__LuaCJK_settings_push:
  }
\prg_new_conditional:Npnn \ctex_if_ccglue_touched: { TF }
  {
    \skip_if_eq:nnTF
      { \l__ctex_ccglue_skip } { \l__LuaCJK_ccglue_skip }
      { \prg_return_false:   } { \prg_return_true: }
  }
\ctex_at_end:n
  {
    \cs_set_protected:Npn \__ctex_update_stretch_auxii:
      {
        \ctex_if_ccglue_touched:TF
          { \ctex_update_ccwd: } { \__ctex_update_stretch_auxiii: }
      }
  }
⟨/ctex⟩
```

The gap between CJK and Latin text has its own default under ctex, `.25em`, and follows the size in the same way.

```
⟨*ctex⟩
\skip_set:Nn \l__LuaCJK_ecglue_skip { .25em plus 1pt minus 1pt }
\skip_new:N \l__ctex_xkanjiskip_skip
\skip_set_eq:NN \l__ctex_xkanjiskip_skip \l__LuaCJK_ecglue_skip
\tl_new:N \l__ctex_xkanjiskip_tl
\tl_set:Nn \l__ctex_xkanjiskip_tl { .25em plus 1pt minus 1pt }
\cs_new_protected:Npn \ctex_update_xkanjiskip:
  {
    \skip_if_eq:nnT
      { \l__LuaCJK_ecglue_skip } { \l__ctex_xkanjiskip_skip }
      {
        \skip_set:Nn \l__ctex_xkanjiskip_skip { \l__ctex_xkanjiskip_tl }
        \skip_set_eq:NN
          \l__LuaCJK_ecglue_skip \l__ctex_xkanjiskip_skip
        \bool_set_true:N \l__LuaCJK_ecglue_touched_bool
        \__LuaCJK_settings_push:
      }
  }
\clist_map_inline:nn { \ctex_update_xkanjiskip: , \ctex_update_kanjisize: }
  { \ctex_add_to_selectfont:n {#1} }
⟨/ctex⟩
```

The ctex option `punct` selects a punctuation style, and `space` has no counterpart here.

```
⟨*ctex⟩
\cs_new_protected:Npn \ctex_set_jfm:n #1
  {
    \tl_set:Nn \l__ctex_jfm_tl {#1}
    \LuaCJKsetup { PunctStyle = {#1} }
  }
\cs_generate_variant:Nn \ctex_set_jfm:n { o }
\tl_new:N \l__ctex_jfm_tl
\ctex_define:n
  {
    space .code:n    =
      { \msg_warning:nn { ctex } { invalid-option } }
  }
\ctex_define:n
  {
    punct .code:n    =
      {
        \tl_set:Ne \l__ctex_punct_tl {#1}
```

```
        \ctex_set_jfm:o { \l__ctex_punct_tl }
      },
    punct .default:n = { quanjiao },
  }
\ctex_define:n
  { experiment .meta:nn = { ctex / experiment } {#1} }
\keys_define:nn { ctex / experiment }
  {
    CJKecglue .code:n           =
      {
        \tl_set:Nn \l__ctex_xkanjiskip_tl {#1}
        \skip_set:Nn \l__ctex_xkanjiskip_skip {#1}
        \ctex_ltj_set_xkanjiskip:N \l__ctex_xkanjiskip_skip
      },
    CJKecglue .value_required:n = true
  }
\ctex_set_jfm:o { \l__ctex_punct_tl }
```
⟨/ctex⟩

A very small part of the luatexja parameter interface is kept, for documents that set the skips directly. Only `kanjiskip` and `xkanjiskip` are understood; any other key is silently ignored, because this is not a reimplementation of the luatexja parameter system.

⟨*ctex⟩
```
\keys_define:nn { LuaCJK / ctex / ltj }
  {
    kanjiskip  .code:n = { \__LuaCJK_set_ccglue:n {#1} },
    xkanjiskip .code:n = { \__LuaCJK_set_ecglue:n {#1} },
    unknown    .code:n = { },
  }
\NewDocumentCommand \ltjsetparameter { m }
  { \keys_set:nn { LuaCJK / ctex / ltj } {#1} }
\cs_new:Npn \ltjgetparameter #1
  {
    \str_case:nnF {#1}
      {
        { kanjiskip  } { \skip_use:N \l__LuaCJK_ccglue_skip }
        { xkanjiskip } { \skip_use:N \l__LuaCJK_ecglue_skip }
      } { 0pt }
  }
```
⟨/ctex⟩

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.


### Symbols

\/ . . . . . 5437, 5444
\\ 15, 98, 156, 732, 1110, 1115, 3254, 5345
\␣ . . . . . 84
\~ . . . . . 5436, 5443

### A

\addCJKfontfeatures *9*, *29*, *71*, *233*, 1354
adjacent commands:
adjacent_marks . . . . . 2932
ADJUST . . . . . 1656
adjust . . . . . 4455
\advance . . . . . 5403, 5409, 5440, 5451
\allocationnumber . . . . . 30, 4651
any commands:
any_cjk_math_enabled . . . . . 3138
apply commands:
apply_check_single . . . . . 3899
\apply_fixed_widths . . . . *155*, *156*
apply_fixed_widths . . . . . 3057
apply_kinsoku . . . . . 2907
attached commands:
attached_box . . . . . 2425

### B

\baselineskip *11*, *30*, *84*, 1223, 1237, 1242
\bfdefault . . . . . *10*, *30*, *81*, 1229, 1231
\bfseries . . . . . *81*
block commands:
block_font . . . . . 2200
\boldmath . . . . . *10*, *30*, *81*, *236*
bool commands:
\bool_gset_false:N . . . . . 651
\bool_gset_true:N . 649, 1592, 1599
\bool_if:NTF 56, 61, 143, 145, 147, 163, 244, 401, 403, 676, 679, 697, 699, 793, 835, 1008, 1145, 1259, 1293, 1336, 1533, 1600, 1632, 4749, 4768, 4770, 5079, 5087, 5145
\bool_lazy_all:nTF . . . . . 4731
\bool_lazy_and:nnTF . . . 1225, 4712
\bool_lazy_or:nnTF . . . . . 937, 947
\bool_new:N . . . . . 39, 40, 110, 111, 112, 151, 223, 381, 383, 451, 452, 455, 458, 471, 472, 724, 725, 726, 896, 897, 898, 899, 979, 1134, 1593, 4663, 4664, 4665, 4666, 4682
\bool_not_p:n . . . . . 4735
\bool_set_eq:NN . 779, 781, 799, 801
\bool_set_false:N . . . . . 135, 136, 137, 238, 562, 583, 606, 645, 808, 929, 1030, 1187, 1297, 5005, 5006, 5007, 5008
\bool_set_true:N . . . . . 47, 52, 119, 124, 129, 165, 242, 392, 394, 566, 587, 603, 641, 788, 855, 866, 919, 920, 921, 922, 1027, 1147, 1295, 4973, 4979, 5004, 5026, 5546, 5552, 5558, 5590
bound commands:
bound_width . . . . . 3288
box commands:
\box_ht:N . . . . . 5122, 5379
\box_ht_plus_dp:N . . . . . 5373
\box_move_up:nn . . . . . 5050, 5132, 5168, 5219
\box_new:N . . . . . 728, 5037, 5336
\box_set_dp:Nn 5068, 5127, 5183, 5378
\box_set_ht:Nn 5067, 5125, 5182, 5376
\box_use_drop:N . . . . . 5069, 5134, 5184, 5382
box_visible . . . . . 2421
\box_wd:N . . . . . 751, 5113

## C

\catcode . . . . . 5449
\ccwd . . . . . 5528, 5530, 5532, 5535
\cdp . . . . . 5528, 5537
cff commands:
  cff_box . . . . . 3523
  cff_dict . . . . . 3452
  cff_index . . . . . 3433
  cff_load . . . . . 3499
  cff_string . . . . . 3448
char commands:
  \char_generate:nn . . . . . 72
  \char_set_catcode_ignore:n . . . . . 1532, 1536, 1538, 1540
  \char_set_catcode_space:n . . . . 83
\cht . . . . . 5528, 5536
\CJKfamily . . . . . *4*, *6*, *23*, *25*, *78*, *88*, *233*, *237*, 1464, 1496, 1497, 1498, 1499, 1500, 1501
\CJKfamilydefault . . . . . *7*, *9*, *26*, *28*, *76*, *85*, *94*, *236*, *239*, 1083, 1089, 1103, 1342, 1617, 1620
CJKfilltwosides (env.) . . . . . 1, 1, 5336
CJKfilltwosides* (env.) . . . . . 5390
\CJKfontspec . . *6*, *26*, *233*, *236*, *240*, 1485
\CJKrmdefault . . . . . *6–9*, *25–28*, *76*, *85*, *86*, *94*, *236*, *237*, *239*, 1095, 1342, 1394, 1624, 1628
\CJKsfdefault *7*, *26*, *88*, 1342, 1399, 1625
\CJKsout . . . . . *21*, *42*, 4828
\CJKttdefault . . *7*, *26*, 1342, 1404, 1626
\CJKunderanyline . . . . *20*, *41*, *233*, 5208
\CJKunderanysymbol *20*, *42*, *203*, *233*, 5307
\CJKunderdblline . . . . *19*, *40*, *233*, 5029
\CJKunderdot . . . . . *20*, *42*, *165*, *167*, *186*, *233*, 5316
\CJKunderline . . *18*, *39*, *203*, 4655, 4816
\CJKunderwave . . . . . *19*, *41*, *233*, 5037
\CJKxout . . . . . *20*, *41*, *233*, 5160
\clap . . . . . *118*, *237*
class commands:
  class_of_char . . . . . 2409
  class_of_node . . . . . 2416
clist commands:
  \clist_const:Nn . . . . . 1525, 1527
  \clist_gput_right:Nn . . . . . 191
  \clist_if_in:NnTF . . . . . 190, 205
  \clist_map_inline:Nn . . . . . 239, 1309, 1531, 1535
  \clist_map_inline:nn . . . . . 22, 201, 313, 874, 927, 999, 1270, 1342, 1502, 5014, 5517, 5528, 5594
  \clist_new:N . . . . . 185, 198
  \clist_put_right:Nn . . . . 206, 1458
  \clist_remove_all:Nn . . . . . 208
\colorlet . . . . . *240*
copy commands:
  copy_attributes . . . . . 4216
  copy_glyph . . . . . 2360
cs commands:
  \cs_generate_variant:Nn . . . . . 483, 484, 905, 1433, 5603
  \cs_gset_eq:NN . . . . . 84, 85, 851
  \cs_gset_protected:Npn . . . . . 852
  \cs_gset_protected:Npx . 822, 843
  \cs_if_exist:NTF . . . . . 18, 820, 841, 849, 1210, 1645, 5346, 5354, 5360, 5383, 5512
  \cs_if_exist_p:N . . . . . 1228
  \cs_new:Npn . 12, 673, 675, 5199, 5642
  \cs_new_eq:NN . *67*, 5524, 5526, 5527
  \cs_new_protected:Npn . . . . . 44, 49, 54, 68, 75, 101, 199, 225, 231, 250, 252, 258, 270, 289, 297, 310, 335, 341, 361, 367, 406, 677, 735, 811, 828, 833, 839, 864, 869, 885, 906, 913, 915, 925, 956, 980, 995, 1024, 1032, 1063, 1081, 1087, 1118, 1143, 1195, 1207, 1217, 1247, 1257, 1291, 1300, 1305, 1359, 1423, 1434, 1439, 1496, 1497, 1498, 1499, 1500, 1501, 1512, 1529, 1543, 1545, 1552, 1560, 1594, 4691, 4693, 4699, 4708, 4710, 4765, 4786, 4814, 4828, 4895, 5000, 5018, 5023, 5041, 5077, 5085, 5090, 5160, 5231, 5233, 5235, 5282, 5294, 5302, 5329, 5341,

5357, 5522, 5525, 5529, 5531, 5533, 5543, 5549, 5555, 5582, 5598
\cs_set_eq:NN ..... 5345
\cs_set_protected:Npn ..... 5569
\cs_set_protected:Npx ..... 1474
\cs_to_str:N ..... 1473, 1589, 1590
\cs_undefine:N ..... 5519
ctex commands:
\ctex_ ..... *209*
\ctex_add_to_selectfont:n ..... 5522, 5595
\ctex_at_end:n ..... 5567
\ctex_define:n ... 5605, 5610, 5619
\ctex_if_ccglue_touched: .. 5561
\ctex_if_ccglue_touched:TF . 5571
\ctex_ltj_set_kanjiskip:N .. 5543
\ctex_ltj_set_xkanjiskip:N .. ..... 5549, 5627
\ctex_ltj_zero_globaldefs: . 5524
\ctex_set_jfm:n ..... ..... 5598, 5603, 5615, 5631
\ctex_update_ccglue: ..... 5555
\ctex_update_ccwd: .... 5529, 5572
\ctex_update_em_unit: ..... 5531
\ctex_update_kanjisize: 5533, 5594
\ctex_update_xkanjiskip: ... ..... 5582, 5594
ctex internal commands:
\l__ctex_ccglue_skip ..... ..... 5541, 5542, 5557, 5564
\l__ctex_jfm_tl ..... 5600, 5604
\l__ctex_punct_tl . 5614, 5615, 5631
__ctex_update_stretch_auxii: ..... 5569
__ctex_update_stretch_-auxiii: ..... 5572
\l__ctex_xkanjiskip_skip 5578, 5579, 5585, 5587, 5589, 5626, 5627
\l__ctex_xkanjiskip_tl ..... ..... 5580, 5581, 5587, 5625
\cwd ..... 5528, 5535

## D

\def 5400, 5405, 5414, 5423, 5428, 5433, 5442, 5446, 5453, 5471, 5476, 5480
\defaultCJKfontfeatures ..... ..... *9*, *29*, *233*, *240*, 1352
dim commands:
\dim_compare:nNnTF . 739, 756, 5351
\dim_compare_p:nNn ..... 4733
\dim_eval:n .... 765, 1045, 1236, 1241, 4851, 4856, 4870, 4875, 4906, 4907, 4922, 4928, 5110, 5118
\dim_gset:Nn ..... 750
\dim_new:N ..... 453, 456, 719, 720, 721, 4655, 4656, 4659, 4667, 4668, 4677, 4678, 4679, 4680, 4684, 4685, 5038, 5039, 5040, 5209, 5337, 5338, 5339, 5528
\dim_set:Nn ..... 454, 457, 567, 588, 737, 754, 4687, 4688, 4689, 4690, 4717, 4718, 4769, 4772, 4781, 4783, 4943, 4947, 4972, 4978, 5010, 5011, 5012, 5013, 5016, 5092, 5093, 5108, 5116, 5214, 5348, 5352, 5372, 5374, 5530, 5532, 5535, 5536, 5537
\dim_set_eq:NN ..... 743
\dim_to_decimal_in_sp:n .... ..... 172, 181, 676, 1044, 4755, 4759, 4760, 4802, 4866, 5251
\dim_use:N ..... ... 764, 4777, 4778, 4791, 4792, 4833, 4834, 4900, 4901, 4915, 4916, 4917, 5120, 5143, 5144, 5220
\dim_zero:N ... 563, 584, 4946, 5002
\c_zero_dim ..... 5067, 5068, 5126, 5128, 5182, 5183, 5537
\directlua ..... 5478
disabled ..... 2168
DISC ..... 1656
dot commands:
dot_box ..... 4310
draw commands:
\draw_begin: ..... 5054, 5172
\draw_end: ..... 5063, 5178
\draw_path_curveto:nnn 5058, 5060
\draw_path_lineto:n ..... 5176
\draw_path_moveto:n ... 5057, 5175
\draw_path_use_clear:n 5062, 5177

\draw_set_linewidth:n . 5056, 5174
drawn . . . . . 4502
drop commands:
drop_orphans . . . . . 2325

E

\edef . . . . . 5477
edge commands:
edge_mark . . . . . 2964
\else . . . . . 5417, 5418, 5472, 5481
\emph . . . . . *238*
environments:
CJKfilltwosides . . . . . 1, 1, 5336
CJKfilltwosides* . . . . . 5390
exp commands:
\exp_args:Nc . . . . . 29, 929, 1302, 1344, 1345, 4801, 4847, 4864, 5262, 5268, 5272
\exp_args:NV 295, 747, 991, 1085, 1104, 1127, 1128, 1151, 1175, 1270, 1317, 1356, 1393, 1398, 1403, 1409, 1445, 1467, 1476, 1491, 1493, 1549, 4822, 4910, 4935, 5074, 5148, 5188, 5226, 5298
\exp_args:Nv . . . . . 1505
\exp_not:N . 825, 845, 1475, 1624, 1625, 1626, 1628, 4905, 4921, 5104, 5106, 5108, 5110, 5113, 5116, 5118, 5121, 5122, 5125, 5126, 5127, 5128, 5129, 5130, 5131, 5132, 5133, 5134, 5135, 5136, 5137, 5217, 5219, 5221, 5305
\exp_not:n . . . . . 305, 308, 824, 846, 942, 952, 1365, 1366, 1475, 4995, 5107, 5221, 5250
\expandafter . . . . . 1301, 5474, 5479
\ExplFileVersion . . . . . 11

F

fallback commands:
fallback_font . . . . . 2245
\familydefault . . . . . *7*, *26*, *239*, 1622
\fangsong . . . . . *6*, *26*, *236*, 1496, 5518
\fi . . . . . 5418, 5420, 5426, 5431, 5475, 5484, 5490, 5494
file commands:
\file_if_exist:nTF . . . . . 1554
\file_input:n . . . . . 1555
find commands:
find_cjk_math . . . . . 3221
first commands:
first_punct . . . . . 2464
first_visible_class . . . . . 2429
\font . . . . . 1232, 1239, 1244
font commands:
font_advance . . . . . 3415
font_box . . . . . 3678
font_cmap . . . . . 3346
font_file . . . . . 3307
font_files . . . . . 3296
font_glyph . . . . . 3383
\fontencoding . . . . . 886
\fontfamily . . . . . 886
\fontseries . . . . . 1221, 1231
\fontshape . . . . . 1222
\fontsize . . . . . 1223, 1235, 1240
\fontspec . . . . . *6*, *26*
fontspec commands:
\fontspec_gset_family:Nnn . . . . . 276, 905, 965, 1373, 1446
forbid commands:
forbid_marked_disc . . . . . 2925
forbids commands:
forbids_break . . . . . 2942
fp commands:
\fp_new:N . . . . . 382
\fp_set:Nn . . . . . 125, 393
\fp_use:N . . . . . 146, 402
free commands:
free_node . . . . . 1656

G

\gdef . . . . . 5438, 5445
\global . . . . . 1232, 1239, 1244, 5483
GLUE . . . . . 1656
glue commands:
glue_width . . . . . 4489
GLYPH . . . . . 1656

group commands:
\group_begin: ..... 70, 77, 82, 274, 714, 744, 854, 964, 997, 1150, 1174, 1219, 1316, 1372, 1444, 1619, 4722, 4818, 4889, 5031, 5153, 5193, 5212, 5287, 5309, 5318, 5331, 5344
\group_end: ..... 73, 80, 86, 281, 717, 753, 857, 974, 1022, 1158, 1182, 1245, 1321, 1382, 1461, 1630, 4727, 4826, 4893, 5035, 5158, 5197, 5229, 5290, 5314, 5326, 5334, 5384

## H

half commands:
half_mark ..... 1964
handle commands:
handle_boundary ..... 2576
handle_source_space ..... 2707
hang commands:
hang_amount ..... 3746
hang_bounds ..... 3818
hang_rule ..... 3759
has commands:
has_attribute ..... 1656
has_char ..... 2231
has_char_cache ..... 2231
hbox commands:
\hbox:n ..... 5052, 5104, 5164, 5170, 5217, 5221, 5328
\hbox_set:Nn . 749, 5048, 5106, 5166
\hbox_to_wd:nn ..... 5045
head commands:
head_lead ..... 3759
\heiti ..... 6, 26, 236, 1496, 5518
\hfill ..... 65, 239
hide commands:
hide_list ..... 4224
hide_text ..... 4261
HLIST ..... 1656
hook commands:
\hook_gput_code:nnn ..... 20, 711, 861, 880, 882, 1333, 1334, 1504, 1507, 1615, 1643, 5514, 5523
\hspace ..... 16, 37, 65, 133, 241
hyphenate commands:
hyphenate_runs ..... 3995

## I

i16 ..... 3298
\IfBooleanT ..... 161
\IfNoValueTF ..... 1473
\ifnum ..... 5439, 5450
\ifx ..... 5472, 5488, 5492
\ignorespacesafterend ..... 5385
inherit commands:
inherit_attributes ..... 2526
INS ..... 1656
insert commands:
insert_boundary_glue ..... 2576
insert_dots ..... 4336
int commands:
\int_compare:nNnTF ..... 773, 805, 982, 1200, 1307
\int_const:Nn ..... 30, 1141, 4651
\int_eval:n ..... 1003
\int_from_hex:n ..... 295
\int_gincr:N ..... 1199, 1488, 4798, 4842, 5257
\int_gset:Nn ..... 1140, 1153, 1177, 1233, 1319, 4725
\int_gzero:N ..... 1312, 5009
\int_new:N ..... 461, 727, 894, 1135, 1138, 1139, 1216, 1304, 1485, 4670, 4686
\int_set:Nn 462, 616, 783, 1250, 1273
\int_step_inline:nnn .. 1537, 1539
\int_use:N ..... 30, 696, 984, 1015, 1156, 1180, 1265, 1266, 1278, 1279, 1286, 1326, 1328, 1490, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1588, 4651, 4654, 4742, 4761, 4800, 4844, 5098, 5204, 5259
\c_zero_int ..... 773, 1130, 1132, 1162, 1169, 1185
is commands:
is_break_point ..... 2998
is_inserted ..... 2521

ITALICCORR . . . . . 1656
item commands:
item_label . . . . . 3759
\itshape . . . . . 238

K

\kaishu . . . . . 6, 26, 236, 1496, 5518
KERN . . . . . 1656
keys commands:
\keys_define:nn . . . . . 115, 384, 487, 4938, 5621, 5634
\keys_if_exist:nnTF . . . . . 4695
\l_keys_key_str . . . . . 663, 4993
\keys_set:nn . . . . . 138, 395, 670, 4692, 4696, 4697, 5641
\l_keys_value_tl . . . . . 4994, 4995
keyval commands:
\keyval_parse:NNn . . 256, 930, 4704
kind . . . . . 4115

L

last commands:
last_punct . . . . . 2482
last_visible_class . . . . . 2446
\lccode . . . . . 5436, 5437, 5443, 5444
\let . . . . . 5421, 5422
LIGATURE . . . . . 1656
\lishu . . . . . 6, 26, 236, 1496, 5518
literal . . . . . 4523
\llap . . . . . 118, 237
LOCAL commands:
LOCAL_PAR . . . . . 1656
\loop . . . . . 5435, 5448
\lowercase . . . . . 5438, 5445
\ltjgetparameter . . . . . 5642
\ltjsetparameter . . . . . 5640
lua commands:
\lua_escape:n . . . . . 142, 172, 180, 195, 400, 685, 701, 1327, 1517
\lua_load_module:n . . . . 1574, 4649
\lua_now:n . . . . . 139, 169, 177, 192, 338, 344, 364, 370, 378, 397, 681, 1012, 1056, 1283, 1323, 1514, 1544, 1575, 1597, 4653, 4739, 4752, 5095, 5201
LuaCJK . . . . . 1648
LuaCJK internal commands:
__LuaCJK_add_fallback:nn . . . . . 1418, 1419, 1421, 1439, 1439
__LuaCJK_add_family_features:n . . . . . 1356, 1359, 1359
\g__LuaCJK_attached_attr . . . . . 22
\c__LuaCJK_attached_attr_int . . . . . 22, 1586
\c__LuaCJK_attached_chars_clist . . . . . 1527, 1535
\l__LuaCJK_auto_fallback_bool . . . . . 441, 641, 645, 699
\l__LuaCJK_autofake_bold_given_bool . . . . . 896, 919, 938
\g__LuaCJK_autofake_bold_tl . 414, 425, 433, 441, 635, 636, 935, 942
\l__LuaCJK_autofake_slant_given_bool . . . . . 896, 920, 948
\g__LuaCJK_autofake_slant_tl 418, 426, 437, 441, 657, 658, 945, 952
\g__LuaCJK_block_fontid_int . . . . . 1304, 1312, 1319, 1328
\l__LuaCJK_block_key_tl . . . . . 217, 1311, 1314
\g__LuaCJK_block_names_clist . . . . . 184, 239, 1309
\g__LuaCJK_block_prop . . . . . 184
\l__LuaCJK_blocks_off_clist . . . . . 198, 205, 206, 208, 701
\l__LuaCJK_bold_font_given_bool . . . . . 896, 921, 939
__LuaCJK_bool_js:N . . . . . 673, 686, 687, 688, 689, 695
\l__LuaCJK_bound_width_warned_bool . . . . . 151, 163, 165
\l__LuaCJK_break_between_puncts_bool . . 441, 603, 606, 697
__LuaCJK_build_features:n . . . . . 275, 908, 913, 925, 1445
__LuaCJK_build_ranges:n . . . . . 188, 310, 337, 343, 363, 369
\l__LuaCJK_ccglue_skip . . . . . 37, 46, 58, 776, 795, 1589, 5542, 5545, 5557, 5564, 5646

\l__LuaCJK_ccglue_touched_bool 39, 47, 56, 780, 799, 5546, 5558
\l__LuaCJK_check_single_tl . . . . . 441, 611, 613, 695
\l__LuaCJK_cj_strict_tl . . . . . 441, 511, 514, 687
\l__LuaCJK_cjk_math_tl . . . . . 441, 520, 522, 689, 1120
\g__LuaCJK_cjk_space_attr . . . . . 22, 492, 493
\c__LuaCJK_cjk_space_attr_int . . . . . 22, 1578
\l__LuaCJK_default_feature_tl . . . . . 896, 932, 1353
__LuaCJK_detect_shaping: . . . . . 1593, 1594, 1613
\g__LuaCJK_disabled_attr . . . . . 22, 707, 709, 784
\c__LuaCJK_disabled_attr_int . . . . . 22, 1585
\l__LuaCJK_ecglue_skip . . . . . 37, 51, 63, 778, 797, 1590, 5551, 5577, 5579, 5585, 5589, 5647
\l__LuaCJK_ecglue_touched_bool 40, 52, 61, 782, 801, 5552, 5590
\g__LuaCJK_embolden_factor_tl . . . . . 415, 432, 441, 634
__LuaCJK_ensure_family_default: . . . 92, 1087, 1087, 1637
__LuaCJK_explicit_nobreak_space: . . . . . 67, 75, 85
__LuaCJK_explicit_space: . . . . . 67, 68, 84
\g__LuaCJK_explicit_space_attr . . . . . 22, 71, 78
\c__LuaCJK_explicit_space_attr_int . . . . . 22, 1588
__LuaCJK_extract_font_keys:nn . . . . . 217, 258, 1436
\l__LuaCJK_fallback_family_tl . . . . . 441, 1447, 1451, 1458
\l__LuaCJK_fallback_ids_tl . . . . . 441, 998, 1004, 1005, 1010, 1016
\l__LuaCJK_fallback_key_tl . . . . . 474, 1414, 1451, 1456, 1460
\g__LuaCJK_fallback_prop . . . . . 441, 985, 1268, 1455, 1459
\g__LuaCJK_fallback_pushed_prop . . . . . 978, 1055
\g__LuaCJK_family_block_prop . . . . . 217, 1313
\l__LuaCJK_family_default_init_tl . . . . . 1342, 1617
__LuaCJK_family_font:n . 747, 885, 885, 1001, 1151, 1175, 1220, 1317
\l__LuaCJK_family_name_tl . . . . . 890, 1148, 1151, 1157, 1163
\g__LuaCJK_family_prop . . . . . 745, 887, 967, 975, 1065, 1084, 1090, 1092, 1096, 1098, 1125, 1148, 1171, 1383, 1634
\l__LuaCJK_family_tl . . . . 745, 887, 975, 985, 990, 1006, 1019, 1067, 1123, 1148, 1157, 1311, 1357
\l__LuaCJK_feature_tl . . . . . 277, 904, 909, 932, 934, 941, 951, 960, 961, 966, 1361, 1363, 1365, 1374, 1386, 1447
\l__LuaCJK_features_tl . . . . . 52, 217, 1437, 1443, 1445
\l__LuaCJK_fntef_any_code_tl . . . . . 4903, 4911, 4919, 4936, 5043, 5075, 5102, 5149, 5162, 5189, 5208, 5215, 5227
\l__LuaCJK_fntef_any_depth_dim . . . . . 5092, 5121, 5144, 5208, 5214, 5220
__LuaCJK_fntef_body:n . 4708, 4708, 4823, 4825, 4892, 5034, 5157, 5196, 5228, 5313, 5321, 5324
\l__LuaCJK_fntef_bottom_dim . . . . . 4655, 4772, 4778, 4784, 4978
\l__LuaCJK_fntef_bottom_given_bool 4655, 4770, 4979, 5007
\l__LuaCJK_fntef_cache_key_tl . . . . . 4671, 4788, 4796, 4810, 4830, 4840, 4883, 4897, 4911, 4912, 4936, 5073, 5075, 5140, 5149, 5187, 5189, 5224, 5227, 5248, 5255, 5278

\l__LuaCJK_fntef_cache_name_-tl . . . . . 4672, 4796, 4799, 4801, 4810, 4812, 4840, 4843, 4848, 4865, 4883, 4885, 5255, 5258, 5263, 5269, 5273, 5278, 5280
__LuaCJK_fntef_capture_-color: . . . . . 5247, 5282, 5282
\l__LuaCJK_fntef_color_key_tl . . . . . 4655, 5239, 5244, 5252
\g__LuaCJK_fntef_color_ops_tl . . . . . 4655, 5289, 5291
\l__LuaCJK_fntef_color_ops_tl . . . . . 4655, 5285, 5291, 5296, 5299
__LuaCJK_fntef_decoration_-setup:n . . . . . 4890, 4938, 5018, 5032, 5154, 5194, 5213, 5310
\l__LuaCJK_fntef_depth_dim . . . . . 4655, 4719, 4769, 4773, 4900, 4906, 4908, 4915, 4924, 4930, 4962, 5010
__LuaCJK_fntef_dots . . 5328, 5332
\l__LuaCJK_fntef_dots_bool . . . . . 4655, 4713, 5008, 5026
\g__LuaCJK_fntef_dots_color_-int . . . . . 4655, 4725, 4761, 5009
\l__LuaCJK_fntef_dots_js_tl . . . . . 4655, 4716, 4730, 4736, 4758
__LuaCJK_fntef_dots_setup:n . . . . . 5023, 5023, 5319
\l__LuaCJK_fntef_dots_size_-dim . . . . . 4655, 4717, 4759
\l__LuaCJK_fntef_dots_top_dim . . . . . 4655, 4718, 4760
\l__LuaCJK_fntef_em_descent_-dim . . . . . 5040, 5093, 5120, 5143
\l__LuaCJK_fntef_fill_axis_-dim . . . . . 5336, 5374, 5377
__LuaCJK_fntef_fill_begin:nn . . . . . 5341, 5388, 5391
\l__LuaCJK_fntef_fill_box . . . . . 5336, 5350, 5366, 5367, 5373, 5376, 5378, 5379, 5382
__LuaCJK_fntef_fill_end: . . . . . 5357, 5389, 5392
\l__LuaCJK_fntef_fill_position_-tl . . . . . 5336, 5347, 5362
\l__LuaCJK_fntef_fill_total_-dim . . . . . 5336, 5372, 5377, 5379
\l__LuaCJK_fntef_fill_width_-dim . . . . . 5336, 5348, 5351, 5352
\g__LuaCJK_fntef_format_int . . . . . 4670, 4798, 4800, 4842, 4844, 5257, 5259
\g__LuaCJK_fntef_format_prop . . . . . 4669, 4795, 4809, 4839, 4882, 5254, 5277
\l__LuaCJK_fntef_format_tl . . . . 4658, 4720, 4723, 4790, 4794, 4820, 4835, 4837, 4838, 4845, 4982, 4983, 5237, 5243, 5266, 5284
__LuaCJK_fntef_formatted_-underline: . . . . 4786, 4786, 4825
\l__LuaCJK_fntef_gap_dim 4679, 4689, 4917, 4926, 4932, 4966, 5012
\l__LuaCJK_fntef_height_dim . . . . . 4680, 4690, 4833, 4853, 4858, 4872, 4877, 4968, 5013
\l__LuaCJK_fntef_hidden_bool . . . . . 4664, 4735, 4749, 4958, 5005
\l__LuaCJK_fntef_hidden_js_tl . . . . . 4662, 4750, 4751, 4757
__LuaCJK_fntef_literal_-direct:n . . . . . 5282, 5298, 5302
__LuaCJK_fntef_option_bare:n . . . . . 4691, 4693, 4705
__LuaCJK_fntef_option_-pair:nn . . . . . 4691, 4691, 4705
__LuaCJK_fntef_options:n . . . . . 4691, 4699, 4767, 5020, 5025
\l__LuaCJK_fntef_pass_tl . . . . . 4655, 4775, 4823, 4991
\l__LuaCJK_fntef_pattern_box . . . . . 5037, 5048, 5067, 5068, 5069, 5106, 5113, 5122, 5125, 5127, 5134, 5166, 5182, 5183, 5184
__LuaCJK_fntef_pattern_-color: . . . 5055, 5173, 5282, 5294
__LuaCJK_fntef_prepare_-underline:n . . . . . 4765, 4819

__LuaCJK_fntef_push_settings: ..... 4710, 4710, 4780, 5021, 5027
__LuaCJK_fntef_reset_decoration: ..... 4701, 5000
\l__LuaCJK_fntef_rule_depth_dim ..... 4668, 4783, 4792, 4807
\l__LuaCJK_fntef_rule_height_dim ..... 4667, 4781, 4791, 4806
\g__LuaCJK_fntef_settings_attr ..... 4650, 4652
\c__LuaCJK_fntef_settings_attr_int ..... 4651, 4654, 4742
__LuaCJK_fntef_silent_begin: ..... 5077, 5131
\l__LuaCJK_fntef_silent_bool . 4655, 4960, 5004, 5079, 5087, 5145
__LuaCJK_fntef_silent_end: ..... 5085, 5135
\l__LuaCJK_fntef_skip_js_tl ..... 4661, 4747, 4748, 4756
\l__LuaCJK_fntef_skip_tl .... 4660, 4734, 4746, 4954, 4955, 5003
__LuaCJK_fntef_slash: 5160, 5195
\l__LuaCJK_fntef_subtract_dim ..... 4659, 4733, 4755, 4943, 4946, 4947, 5002
\l__LuaCJK_fntef_symbol_kern_dim ..... 5038, 5108, 5130, 5137
\l__LuaCJK_fntef_symbol_raise_dim ..... 5039, 5116, 5133
\l__LuaCJK_fntef_symbol_sep_dim ..... 4986, 5016
\l__LuaCJK_fntef_symbol_tl ..... 4681, 4714, 4985, 5107, 5146, 5155, 5311, 5320
\l__LuaCJK_fntef_textformat_tl ..... 4655, 4709, 4988
\l__LuaCJK_fntef_thickness_dim ..... 4655, 4773, 4834, 4854, 4859, 4873, 4878, 4901, 4908, 4916, 4925, 4931, 4964, 5011
\l__LuaCJK_fntef_top_dim ..... 4655, 4769, 4777, 4782, 4972
\l__LuaCJK_fntef_top_given_bool ..... 4655, 4768, 4973, 5006
__LuaCJK_fntef_underdot:n ..... 5321, 5328, 5329
__LuaCJK_fntef_underLine:nn ..... 4814, 4822
__LuaCJK_fntef_use_double: ..... 4895, 4895, 5033
__LuaCJK_fntef_use_drawn_type:nn *194*, *198*, 5074, 5188, 5233
__LuaCJK_fntef_use_leader_type:nn ..... *194*, 4910, 4935, 5148, 5226, 5231, 5231
__LuaCJK_fntef_use_sout: ..... 4828, 4891
__LuaCJK_fntef_use_symbol:n ..... 5090, 5156, 5312, 5323
__LuaCJK_fntef_use_type:nnn ..... 5232, 5234, 5235
__LuaCJK_fntef_wave: . 5041, 5156
\g__LuaCJK_font_attr ..... 22, 1159, 1162, 1640
\c__LuaCJK_font_attr_int ..... 22, 1577, 5098, 5204
\g__LuaCJK_font_feature_prop ..... 896, 909, 960, 1361, 1385
\g__LuaCJK_font_name_prop ..... 896, 910, 958, 1368
__LuaCJK_font_option:n 250, 256
__LuaCJK_font_option:nn 231, 256
\g__LuaCJK_font_shift_prop ..... 441, 971, 1032, 1379, 1452
\g__LuaCJK_fontid_int ..... 982, 984, 1015, 1134, 1153, 1156, 1159, 1307, 1326, 1640
\l__LuaCJK_fontid_tl ..... 441, 984, 988, 1019
\g__LuaCJK_fontspec_int ..... 1485, 1488, 1490
\l__LuaCJK_fontspec_tl ..... 890, 965, 967, 970, 1373, 1378, 1384
__LuaCJK_fv_patch: ..... 828, 839, 862, 863
__LuaCJK_fv_prep_hook: ..... *69*, 828, 833, 845
__LuaCJK_fv_use_verbatim: ..... 851, 856

\l__LuaCJK_hex_int_tl 284, 294, 305
__LuaCJK_hex_store:n .. 289, 303
\l__LuaCJK_hex_tl ..... ..... 284, 291, 292, 293, 295
\g__LuaCJK_inserted_attr ..... 22
\c__LuaCJK_inserted_attr_int ..... 22, 1580
\l__LuaCJK_is_block_bool .... ..... 223, 238, 242, 244
\l__LuaCJK_italic_font_given_bool ..... 896, 922, 949
\l__LuaCJK_latin_punct_tl ... ..... 447, 448, 594, 596, 688
__LuaCJK_load_cj: 515, 1543, 1543
__LuaCJK_load_config_file:n ..... 1549, 1552
__LuaCJK_load_fandol: ..... ..... 1560, 1560, 1635
\g__LuaCJK_load_fandol_bool . ..... 441, 649, 651, 1632
__LuaCJK_load_local_config: ..... 1545, 1545, 1642
\l__LuaCJK_local_config_tl .. .... 441, 621, 622, 623, 1547, 1549
\g__LuaCJK_lua_loaded_bool .. ..... 679, 978, 1259, 1592
\l__LuaCJK_math_fallback_ids_tl .. 1256, 1262, 1263, 1276, 1287
__LuaCJK_math_fallbacks_push:n ..... 1128, 1256, 1257
__LuaCJK_math_family:n .... ..... 1190, 1195, 1249, 1272
__LuaCJK_math_family_fonts:nn ..... 1216, 1217, 1253, 1275
\g__LuaCJK_math_family_int .. ..... 1139, 1199, 1200, 1203
\c__LuaCJK_math_family_max_int ..... 1139, 1200
\g__LuaCJK_math_family_prop . ..... 1139, 1197, 1202, 1251, 1274
\l__LuaCJK_math_family_tl ... ..... 1122, 1124, 1136, 1482
\g__LuaCJK_math_font_attr ... ... 22, 1130, 1132, 1254, 1266, 1286
\c__LuaCJK_math_font_attr_int ..... 22, 1581
__LuaCJK_math_font_update: . ..... 1029, 1118, 1118, 1165, 1296
__LuaCJK_math_size_set: ... ..... 1206, 1207, 1252, 1261
\l__LuaCJK_math_size_tl .... ..... 1206, 1223, 1236, 1241
\g__LuaCJK_math_textfont_int ..... *81*, 1216, 1233, 1265, 1278
__LuaCJK_math_version_refresh: ... *77*, 1291, 1291, 1303
\g__LuaCJK_nobreak_attr .... ..... 22, 715, 787, 804
\c__LuaCJK_nobreak_attr_int . ..... 22, 1583
__LuaCJK_nobreak_space: ... ..... 67, 67, 79
\g__LuaCJK_normal_spaced_chars_tl ..... 1509, 1517, 1522
__LuaCJK_note_feature_key:n ..... 914, 915, 931
__LuaCJK_note_feature_key:nn ..... 913, 931
\l__LuaCJK_option_key_tl .... 217
\l__LuaCJK_option_seq ..... 217
__LuaCJK_parse_font_options:n ..... 217, 252, 261, 1441
\g__LuaCJK_pending_block_prop 217
\l__LuaCJK_punct_bound_width_dim ..... 441, 584, 588, 694
\l__LuaCJK_punct_bound_width_given_bool ... 441, 583, 587, 693
\l__LuaCJK_punct_compress_bool ..... 381, 386, 392, 401
\l__LuaCJK_punct_compress_given_bool ... 110, 119, 135, 143
\l__LuaCJK_punct_compress_value_tl ..... 113, 120, 144
\l__LuaCJK_punct_family_tl .. ... 574, 575, 1136, 1168, 1171, 1181
\g__LuaCJK_punct_font_attr .. ..... 22, 1169, 1183, 1185
\c__LuaCJK_punct_font_attr_int ..... 22, 1582

\g__LuaCJK_punct_fontid_int . . . . . 1136, 1177, 1180, 1183
\l__LuaCJK_punct_keep_bool . . . . . 381, 388, 394, 403
\l__LuaCJK_punct_keep_given_bool . . . . . 110, 129, 137, 147
\l__LuaCJK_punct_keep_value_tl . . . . . 114, 130, 148
\l__LuaCJK_punct_kinsoku_tl . . . . . 441, 505, 507, 686
\l__LuaCJK_punct_ratio_fp . . . . . 125, 146, 381, 387, 393, 402
\l__LuaCJK_punct_ratio_given_bool . . . . . 110, 124, 136, 145
\g__LuaCJK_punct_style_prop . . . . . 87, 103, 396
\l__LuaCJK_punct_style_tl . . . . . 94, 104, 107, 685
\l__LuaCJK_punct_width_dim . . . . . 441, 563, 567, 691
\l__LuaCJK_punct_width_given_bool . . . . . 441, 562, 566, 691
__LuaCJK_push_block_fonts: . . . . . 1167, 1304, 1305
__LuaCJK_push_class:nn . . . . . 323, 324, 330, 331, 332, 341
__LuaCJK_push_fallback: . . . . . 978, 980, 1028, 1166
__LuaCJK_push_fallback_aux:n . . . . . 991, 995
__LuaCJK_push_normal_space:n . . . . . 329, 335
__LuaCJK_push_punct:nn . . . . . 325, 326, 327, 328, 351, 352, 353, 354, 355, 356, 357, 358, 361
__LuaCJK_push_shift:nnn . . . . . 1006, 1032, 1032, 1155, 1179
__LuaCJK_put_feature:n . . . . . 225, 247, 251
__LuaCJK_range_entry:n 297, 313
\l__LuaCJK_range_pair_tl . . . . . 284, 300, 304, 308
\l__LuaCJK_range_seq 284, 299, 301
\l__LuaCJK_ranges_tl . . . . 189, 195, 284, 307, 312, 339, 345, 365, 373
__LuaCJK_refresh_fallbacks: . . . . . 642, 978, 1024, 1462
__LuaCJK_refresh_glues: 54, 1338
__LuaCJK_register_block_fonts:n . . . . . 268, 270
__LuaCJK_register_family:n . . . . . 911, 956, 956
__LuaCJK_register_math_family:n . . . . . 1127, 1247, 1247
\l__LuaCJK_registered_name_tl 895, 958, 959, 966, 1368, 1369, 1374
\c__LuaCJK_removed_chars_clist . . . . . 1525, 1531
__LuaCJK_reset_family: . . . . . 1081, 1081, 1508, 1638
\l__LuaCJK_selectfont_bool . . . . . 1027, 1030, 1134, 1145, 1147, 1187, 1293, 1295, 1297, 1336
__LuaCJK_set_autofake:nn . . . . . 406, 406, 626, 629
__LuaCJK_set_block_enabled:nn . . . . . 199, 214, 216
__LuaCJK_set_ccglue:n . . . . . 44, 44, 489, 5636
__LuaCJK_set_charset:nnn . . . . . 367, 367, 535, 537, 539, 541, 543, 545, 547, 549, 551, 553, 555, 557
__LuaCJK_set_ecglue:n . . . . . 44, 49, 490, 5637
__LuaCJK_set_family:nnn . . . . . 887, 906, 1433, 1437
__LuaCJK_set_family_with_options:nnnn . . . . . *86*, 1393, 1398, 1403, 1409, 1423, 1423, 1476, 1481, 1491
__LuaCJK_set_family_with_options_aux:nnn . . . . . 1428, 1429, 1431, 1434
__LuaCJK_set_normal_spaced_chars: . . . . 379, 1512, 1523, 1612
__LuaCJK_set_punct_style:n . . . . . 94, 101, 501
\g__LuaCJK_settings_attr . . . . . 22
\c__LuaCJK_settings_attr_int . . . . . 22, 1584

__LuaCJK_settings_push: ... 210, 671, 673, 677, 803, 1339, 1611, 5547, 5553, 5559, 5591
__LuaCJK_setup_format_chars: ... 1525, 1529, 1614
\g__LuaCJK_shaping_bool ... 1533, 1593, 1599, 1600
\g__LuaCJK_shift_decl_prop .. ... 262, 441, 962, 1370
\l__LuaCJK_shift_dim_tl ... 441, 962, 963, 968, 972, 1032, 1370, 1371, 1375, 1380
\l__LuaCJK_shift_fid_tl 441, 1032
\l__LuaCJK_shift_key_tl . 441, 970, 972, 1032, 1377, 1380, 1450, 1453
\l__LuaCJK_shift_old_tl 441, 1032
\g__LuaCJK_shift_pushed_prop ... 441, 1032
\l__LuaCJK_shift_tl ... 52, 217, 1442, 1448, 1453
\l__LuaCJK_shift_value_tl ... ... 441, 1032
\g__LuaCJK_slant_factor_tl .. ... 419, 436, 441, 656
__LuaCJK_switch_family:n ... ... 70, 1063, 1063, 1085, 1104, 1467, 1475, 1493, 1505
\l__LuaCJK_tmpa_int ... 894, 1250, 1253, 1254, 1273, 1275, 1279
\l__LuaCJK_tmpa_tl ... ... 276, 280, 745, 747, 890, 985, 991, 1083, 1084, 1085, 1089, 1090, 1102, 1123, 1124, 1125, 1128, 1172, 1175, 1181, 1268, 1270, 1314, 1317, 1408, 1410, 1456, 1457, 1458, 1460, 1466, 1467, 1472, 1475, 1477, 1489, 1492, 1493
\l__LuaCJK_tmpb_tl 893, 988, 989, 990, 1002, 1004, 1006, 1095, 1096, 1099, 1102, 1103, 1104, 1125, 1127
__LuaCJK_toks_put_right:Nn . ... 1300, 1300, 1302
\g__LuaCJK_unknown_family_seq ... 1071, 1073, 1078
__LuaCJK_update_font: ... ... 73, 77, 84, 211, 520, 522, 576, 1134, 1143, 1333, 1483, 1639
\l__LuaCJK_verb_addon_bool .. ... 719, 788, 793, 808
__LuaCJK_verb_addon_compute: ... 735, 786
\l__LuaCJK_verb_box . 719, 749, 751
\l__LuaCJK_verb_cjk_dim ... ... 719, 743, 750, 755
\l__LuaCJK_verb_gap_dim ... ... 719, 754, 756, 764, 765
\l__LuaCJK_verb_mode_tl . 441, 526, 528, 530, 532, 813, 816, 830, 871
\l__LuaCJK_verb_saved_ccglue_-skip ... 719, 775, 796
\l__LuaCJK_verb_saved_ccglue_-touched_bool ... 719, 779, 800
\l__LuaCJK_verb_saved_disabled_-int ... 719, 783, 805
\l__LuaCJK_verb_saved_ecglue_-skip ... 719, 777, 798
\l__LuaCJK_verb_saved_ecglue_-touched_bool ... 719, 781, 802
\l__LuaCJK_verb_space_dim ... ... 719, 737, 743, 755
__LuaCJK_verbatim@font_hook: ... 811, 811, 825, 837
\l__LuaCJK_verbatim_block_-bool ... 441, 835, 855, 866
__LuaCJK_verbatim_block_-disable: ... 828, 828, 836, 867
__LuaCJK_verbatim_env_begin: ... 864, 864, 881
__LuaCJK_verbatim_env_end: . ... 864, 869, 883
\c__LuaCJK_version_tl ... 11, 12
\l__LuaCJK_widow_penalty_int ... 441, 616, 696
__LuaCJK_width_js:Nn 675, 690, 692
\g__LuaCJK_xecglue_attr ... ... 22, 497, 498
\c__LuaCJK_xecglue_attr_int . ... 22, 1579

LuaCJK.attr commands:
LuaCJK.attr_cjk_space . . . . . 1681
LuaCJK.attr_disabled . . . . . 2166
LuaCJK.attr_explicit_space . 1681
LuaCJK.attr_font . . . . . 1681
LuaCJK.attr_inserted . . . . . 1681
LuaCJK.attr_math_font . . . . . 1681
LuaCJK.attr_settings . . . . . 1690
LuaCJK.attr_xecglue . . . . . 1681
LuaCJK.cj . . . . . 1783
LuaCJK.cj commands:
LuaCJK.cj_count . . . . . 1783
LuaCJK.cj_loaded . . . . . 1783
LuaCJK.declare commands:
LuaCJK.declare_ranges . . . . . 1985
LuaCJK.define commands:
LuaCJK.define_punct_style . . 2095
LuaCJK.hpack . . . . . 3987
LuaCJK.hyphenate . . . . . 4019
LuaCJK.is commands:
LuaCJK.is_attached . . . . . 1903
LuaCJK.is_cjk . . . . . 1825
\LuaCJK.is_cjk . . . . . *104*, *118*, *207*
LuaCJK.new commands:
LuaCJK.new_settings . . . . . 1710
LuaCJK.pre commands:
LuaCJK.pre_linebreak . . . . . 3983
LuaCJK.pre_mlist_to_hlist . . 3991
LuaCJK.process . . . . . 3965
LuaCJK.punct commands:
LuaCJK.punct_codes . . . . . 2075
LuaCJK.punct_styles . . . . . 2056
LuaCJK.set commands:
LuaCJK.set_charset . . . . . 2034
LuaCJK.settings . . . . . 1690
LuaCJK.settings commands:
LuaCJK.settings_by_signature 1690
LuaCJK.shaping . . . . . 1679
LuaCJK.warned commands:
LuaCJK.warned_cjk_math . . . . 3219
\LuaCJKCancelSubCJKBlock . . . . . 4, 7, 23, 27, 235, 198
\LuaCJKDeclareCharClass . . . . . 5, 7, 13, 14, 24, 26, 34, 56, 234, 284
\LuaCJKDeclarePunctClass . . . . . 5, 14, 24, 35, 234, 347
\LuaCJKDeclarePunctStyle . . . . . 15, 21, 35, 43, 48, 108, 234, 381
\LuaCJKDeclareSubCJKBlock . . . . . 5, 7, 24, 26, 235, 184
\LuaCJKdisable . . . . . *16*, *37*, *235*, 706, 806, 814, 817, 831, 5493
\LuaCJKEditPunctStyle *14*, *35*, *235*, 110
\LuaCJKemdescent *20*, *41*, *235*, 4719, 5199
\LuaCJKenable . . . . . *16*, *37*, *235*, 706, 785, 807, 872
LuaCJKfntef commands:
\LuaCJKfntef_use_symbol:n . . . *203*
LuaCJKfntef.attr . . . . . 4047
LuaCJKfntef.color commands:
LuaCJKfntef.color_attr . . . . 4047
LuaCJKfntef.dot commands:
LuaCJKfntef.dot_attr . . . . . 4047
LuaCJKfntef.em commands:
LuaCJKfntef.em_descent . . . . 4069
LuaCJKfntef.hpack . . . . . 4626
LuaCJKfntef.new commands:
LuaCJKfntef.new_settings . . . 4101
LuaCJKfntef.settings . . . . . 4047
\LuaCJKnobreak . . . . . *16*, *22*, *37*, *44*, *132*, *133*, *235*, *236*, *241*, 712
\LuaCJKOffVerbAddon . . . . . *17*, *38*, *65*, *235*, *239*, 719
\LuaCJKResetCharClass *14*, *35*, *234*, 376
\LuaCJKRestoreSubCJKBlock . . . . . 4, 7, 23, 27, 235, 198
\LuaCJKsetkern . . *5*, *15*, *24*, *35*, *235*, 175
\LuaCJKsetup . . . . . *4*, *23*, *58*, *64*, *68*, 441, 759, 762, 769, 5349, 5601
\LuaCJKsetwidth . . . . . 5, 12, 15, 24, 32, 35, 235, 151
\LuaCJKShipoutHook . . *17*, *37*, *235*, 710
\LuaCJKVerbAddon . . . . *17*, *38*, *235*, 719
\LuaCJKversion . . . . . *4*, *23*, *235*, 11
luaul commands:
\luaul_apply_color: . . . . . 4724, 4804, 4868, 5274, 5288
\luaul_context_color: . . . . . 4802, 4866, 5245

\luaul_set_color:n ..... 4723, 4794, 4838, 5243

**M**

M ..... 1656
M.block commands:
M.block_fonts ..... 2176
M.block_order ..... 2176
M.blocks ..... 2176
M.charsets ..... 2009
M.fallback ..... 2219
M.math commands:
M.math_fallback ..... 3153
M.yoffset ..... 2225
make commands:
make_glue ..... 2532
make_kern ..... 2566
\makeatletter ..... 5396
\makeatother ..... 5495
MARK ..... 1656
mark commands:
mark_blank ..... 3715
MATH ..... 1656
MATH commands:
MATH_CHAR ..... 3135
MATH_TEXT_CHAR ..... 3135
math commands:
math_family_for ..... 3159
\mbox ..... *240*
\mddefault ..... *10*, *30*, *81*, 1221
mode commands:
\mode_leave_vertical: ..... 5343
msg commands:
\msg_critical:nn ..... 7, 19, 21, 4645, 5503, 5513, 5515
\msg_error:nn ..... 1201
\msg_new:nnn .... 5, 13, 96, 152, 315, 317, 666, 729, 1079, 1108, 1113, 1190, 1558, 1603, 4643, 5501
\msg_new:nnnn ..... 5506
\msg_warning:nn ..... 166, 758, 1093, 1601, 5608
\msg_warning:nnn ..... 106, 333, 359, 662, 1074, 1389, 1556
\msg_warning:nnnn ..... 1101

**N**

new commands:
new_node ..... 1656
\newattribute ..... 29, 4650
\newCJKfontfamily *6*, *25*, *233*, *236*, 1470
\NewDocumentCommand .... 133, 159, 175, 186, 213, 215, 319, 347, 376, 390, 668, 706, 708, 710, 712, 771, 791, 1352, 1354, 1391, 1396, 1401, 1406, 1412, 1464, 1470, 1479, 1486, 1520, 4816, 4887, 5029, 5151, 5191, 5210, 5307, 5316, 5640
\NewDocumentEnvironment .. 5387, 5390
\newfontfamily ..... *6*, *25*
\newif ..... 5397, 5398
\newunderlinetype ..... 4801, 4847, 4864, 5262, 5268, 5272, 5328
next commands:
next_glyph ..... 2915
next_visible ..... 2982
no commands:
no_break_mark ..... 2907
\nobreak ..... *238*
\nobreakspace ..... *67*, 85
node commands:
node_width ..... 4123
\normalfont ..... *236*
\normalspacedchars ..... .... *11*, *14*, *31*, *34*, *35*, *56*, *235*, 1509

**O**

open commands:
open_hang ..... 3759

**P**

\parbox ..... *12*, *32*, *204*, *237*, *240*
PENALTY ..... 1656
\pingfang ..... 5518
placeholder ..... 2285
pre commands:
pre_shaping ..... 2289
prg commands:
\prg_do_nothing: . 5524, 5526, 5527
\prg_new_conditional:Npnn .. 5561
\prg_return_false: ..... 5565

\prg_return_true: . . . . . 5565
\ProcessKeyOptions . . . . . 1610
prop commands:
\prop_gclear:N . . . 260, 1026, 1055
\prop_get:NnNTF . . . . . 483, 745, 958, 960, 962, 985, 987, 1036, 1048, 1125, 1148, 1171, 1268, 1313, 1361, 1368, 1370, 1455, 4795, 4839, 5254
\prop_gput:Nnn . . . . . 88, 89, 90, 91, 92, 93, 189, 245, 262, 279, 396, 484, 909, 910, 967, 971, 1018, 1053, 1202, 1379, 1452, 1459, 4809, 4882, 5277
\prop_if_empty:NTF 263, 1092, 1634
\prop_if_in:NnTF . . . . . 103, 975, 1065, 1084, 1090, 1096, 1197
\prop_item:Nn . . . . . 1251, 1274
\prop_map_break: . . . . . 1099
\prop_map_inline:Nn . . . . 272, 1098
\prop_new:N . . . . . 87, 184, 217, 218, 470, 475, 476, 477, 887, 900, 901, 978, 1142, 4669
\prop_put:Nnn . . . . . 1383, 1385
PUNCT commands:
PUNCT_CLASS . . . . . 1915
punct commands:
punct_class . . . . . 1968
punct_style . . . . . 2103

Q

\quad . . . . . *16*, *37*, *65*, *133*, *241*

R

\raisebox . . . . . *123*
\relax . . . . . 5434, 5439, 5443, 5444, 5447, 5449, 5450, 5472, 5479
\repeat . . . . . 5441, 5452
replace commands:
replace_pair . . . . . 4368
\RequirePackage . . . . . 10, 1644, 1646, 4648, 5395, 5516
restore commands:
restore_source_spaces . . . . . 2692
rewrite commands:
rewrite_fonts . . . . . 2378
rewrite_math . . . . . 3151
rewrite_math_list . . . . . 3151
rewrite_noad . . . . . 3175
\rlap . . . . . *118*, *237*
\rmdefault . . . . . 1624
\rule . . . . . *20*, *41*
run commands:
run_mark . . . . . 2938

S

scale commands:
scale_ink . . . . . 4530
scan commands:
\scan_stop: . . . . . 71, 78
\scriptfont . . . . . 1239
\scriptscriptfont . . . . . 1244
segments . . . . . 4130
\selectfont . . . . . 748, 976, 1001, 1068, 1152, 1176, 1224, 1231, 1238, 1243, 1318, 1387
seq commands:
\seq_gput_right:Nn . . . . . 1073
\seq_if_in:NnTF . . . . . 1071
\seq_map_inline:Nn . . . . . 301
\seq_new:N . . . . . 222, 285, 1078
\seq_set_split:Nnn . . . . . 299
set commands:
set_attribute . . . . . 1656
set_glue . . . . . 2550
\setCJKfallbackfamilyfont . . . . . *8*, *27*, *234*, 1412
\setCJKfamilyfont . . . . . *4*, *6*, *13*, *23*, *25*, *33*, *86*, *233*, *239*, 1391, 1567, 1569, 1571, 1572
\setCJKmainfont . . . . . *4*, *5*, *23*, *25*, *85*, *86*, *233*, 1111, 1391, 1562
\setCJKmathfont *10*, *30*, *78*, *141*, *235*, 1479
\setCJKmonofont . . . . . *4*, *5*, *23*, *25*, *233*, 1391, 1566
\setCJKsansfont . . . . . *4*, *5*, *23*, *25*, *233*, 1391, 1564
settings commands:
settings_at . . . . . 4159

settings_of ..... 1771
\sfdefault ..... 1625
\shapedefault ... *10*, *30*, *81*, 1222, 1229
single commands:
single_rule ..... 4170
skip commands:
\skip_horizontal:n .... 5129, 5136
\skip_if_eq:nnTF ..... 5563, 5584
\skip_new:N ..... ..... 37, 38, 722, 723, 5541, 5578
\skip_set:Nn ..... 41, 42, 46, 51, 58, 63, 5577, 5587, 5626
\skip_set_eq:NN ..... ..... 775, 777, 795, 797, 5542, 5545, 5551, 5557, 5579, 5588
\skip_use:N ..... 5646, 5647
\songti ..... *6*, *26*, *236*, 1496, 5518
SPACESKIP ..... 1656
spacing ..... 2576
str commands:
\c_backslash_str ..... 1511
\str_case:nn ..... 917, 5362
\str_case:nnTF ..... .... 321, 349, 408, 619, 1622, 5644
\str_if_eq:nnTF ..... ..... 203, 235, 241, 412, 424, 430, 560, 573, 581, 674, 1596, 4746, 4942, 4945, 4953, 5247, 5260
\str_if_eq_p:nn .. 1226, 4734, 4736
SUB commands:
SUB_MLIST ..... 3149
suspended commands:
suspended_break ..... 2737
sys commands:
\sys_if_engine_luatex:TF .... ..... 3, 4641, 5499

## T

tag commands:
\tag_mc_begin:n ..... 5354
\tag_mc_end: ..... 5360
\tagpdfparaOff ..... 5346
\tagpdfparaOn ..... 5383
target commands:
target_font ..... 2260
TEX and LATEX 2ε commands:
\@empty ..... 5488, 5492
\@ne ..... 5403, 5409, 5440, 5451
\@normalcr ..... *204*
\@tempcnta .... 5434, 5436, 5437, 5439, 5440, 5447, 5449, 5450, 5451
\addCJKfontfeatures ..... *9*, *29*
\begin ..... *12*, *32*
\char ..... *11*, *31*, *91*
\check@mathfonts ..... *204*
\CJKfamily ..... *6*, *25*
\CJKfontspec ..... *6*, *26*
\CJKsout ..... *21*, *42*
\CJKunderanyline ..... *20*, *41*
\CJKunderanysymbol ..... *20*, *42*
\CJKunderdblline ..... *19*, *40*
\CJKunderdot ..... *20*, *42*
\CJKunderline ..... *18*, *39*
\CJKunderwave ..... *19*, *41*
\CJKxout ..... *20*, *41*
\cleaders ..... *165*, *194*
\color ..... *187*
\CTEX@alchar ..... 5525
\CTEX@beginallalchar ..... 5526
\CTEX@endallalchar ..... 5527
\current@color ..... *201*, *202*
\defaultCJKfontfeatures ... *9*, *29*
\edef ..... *47*
\emph ..... *9*, *28*
\end ..... *12*, *32*
\every@math@size ..... *83*
\f@size .. *80*, 1209, 5530, 5532, 5536
\fi ..... *206*
\FV@FormattingPrep .. 841, 843, 846
\FV@UseVerbatim .... 849, 851, 852
\FVC@Verb ..... *68*
\glb@currsize . *80*, 1210, 1212, 1213
\glb@settings ..... *83*
\hbox ..... *194*, *195*
\hfil ..... *127*
\hfill ..... *127*
\hpack ..... *162*
\hskip ..... *127*
\hss ..... *194*

\ifLuaCJK@listings@cjkrun . . . . . 5397, 5415, 5424, 5429
\ifLuaCJK@listings@defined . . . . . 5398, 5481
\itshape . . . . . 9, 28
\kern . . . . . 237
\lccode . . . . . 206
\leaders . . . . . 20, 41, 165, 170, 181, 187, 194, 195, 199, 200
\leftskip . . . . . 158
\lpcode . . . . . 144
\lst@AddToHook . . . 5399, 5487, 5491
\lst@Append . . . . . 206, 5404, 5410
\lst@AppendLetter . . . . . 5421, 5423
\lst@AppendOther . . . . . 5422, 5428
\lst@ifletter . . . . . 5418
\lst@inlinechars . . . . . 5488, 5492
\lst@length . . . . . 5403, 5409
\lst@letterfalse . . . . . 5412, 5430
\lst@lettertrue . . 5416, 5418, 5425
\lst@Output . 5411, 5418, 5425, 5430
\lst@OutputOther . . . . . 5418
\lst@whitespacefalse . . 5401, 5406
\lstinline . . . . . 18, 39, 209
\LuaCJK@listings@activate . . . . . 5480, 5489
\LuaCJK@listings@catcodes . . . . . 5446, 5485
\LuaCJK@listings@cjkrunfalse . . . . . 5399, 5407, 5413, 5425, 5430
\LuaCJK@listings@cjkruntrue 5419
\LuaCJK@listings@codes 5477, 5479
\LuaCJK@listings@define 5433, 5482
\LuaCJK@listings@definedtrue 5483
\LuaCJK@listings@definepunct . . . . . 5442, 5473
\LuaCJK@listings@letter 5421, 5427
\LuaCJK@listings@other 5422, 5432
\LuaCJK@listings@process . . . . . 5400, 5438
\LuaCJK@listings@processpunct . . . . . 5405, 5445
\LuaCJK@listings@punctuate . . . . . 5471, 5474, 5479
\LuaCJK@listings@punctuation . . . . . 5476, 5486
\LuaCJK@listings@ranges . . . . . 5453, 5482, 5485
\LuaCJK@listings@setcjkflag . . . . . 5402, 5408, 5414
\LuaCJKCancelSubCJKBlock . . . 7, 27
\LuaCJKDeclareCharClass . . 14, 34
\LuaCJKDeclarePunctClass . . 14, 35
\LuaCJKDeclarePunctStyle . . 15, 35
\LuaCJKDeclareSubCJKBlock . . 7, 26
\LuaCJKdisable . . . . . 16, 37
\LuaCJKEditPunctStyle . . . . 14, 35
\LuaCJKenable . . . . . 16, 37
\LuaCJKnobreak . . . . . 16, 37
\LuaCJKOffVerbAddon . . . . . 17, 38
\LuaCJKRestoreSubCJKBlock . . 7, 27
\LuaCJKsetkern . . . . . 15, 35
\LuaCJKsetup . . . . . 4, 23
\LuaCJKsetwidth . . . . . 15, 35
\LuaCJKShipoutHook . . . . . 17, 37
\math@version . . . . . 1226
\mathchoice . . . . . 80
\mathversion . . . . . 10, 30, 83
\mbox . . . . . 233
\newCJKfontfamily . . . . . 6, 25
\nobreak . . . . . 15, 36
\nobreakspace . . . 11, 31, 46, 47, 127
\normalfont . . . . . 7, 26, 76, 85, 94
\normalspacedchars . . . . . 11, 31
\p@ . . . . . 5530, 5532, 5536
\parindent . . . . . 157
\phantom . . . . . 237
\protrudechars . . . . . 144
\raise . . . . . 198
\rightmarginkern . . . . . 144
\rmfamily . . . . . 7, 26, 85
\rpcode . . . . . 144
\scan_stop . . . . . 47
\selectfont . . . . . 70, 78
\setCJKfallbackfamilyfont . . 8, 27
\setCJKfamilyfont . . . . . 6, 25
\setCJKmainfont . . . . . 5, 25
\setCJKmathfont . . . . . 10, 30
\sffamily . . . . . 7, 26, 88

\space . . . . . 47
\text . . . . . 10, 30, 80, 238
\the . . . . . 83
\ttfamily . . . . . 7, 26
\Umathaxis . . . . . 204
\underLine . . . . . 22, 43
\UseVerb . . . . . 17, 38, 238
\UseVerbatim . . . . . 17, 38, 238
\Verb . . . . . 17, 38, 68, 238
\verb . . . . . 17, 38, 68, 70, 238
\verbatim@font . . . . . 68, 69, 238, 820, 822, 824
\VerbatimInput . . . . . 17, 38, 238
\vtop . . . . . 204
\write . . . . . 47
\XeTeXglyphbounds . . . . . 144
\xleaders . . . . . 165, 181, 194
\xout . . . . . 20, 41, 198
tex commands:
\tex_attribute:D . . . . . 5098, 5204
\tex_baselineskip:D . . . . . 41, 59
\tex_currentgrouplevel:D . . . . 773
\tex_endinput:D . . . . 8, 4646, 5504
\tex_expanded:D . . . . . 5304
\tex_font:D . . . . . 738, 740, 741, 1003, 1154, 1178, 1234, 1320
\tex_fontcharwd:D . . . . . 741
\tex_fontdimen:D . . . . . 738, 740
\tex_fontid:D . . . . . 1003, 1154, 1178, 1234, 1320
\tex_hsize:D . . . . . 5352
\tex_hss:D . . . . . 5047, 5070
\tex_ignorespaces:D 1468, 1494, 5355
\tex_leaders:D . . . . . 4805, 4850, 4869, 4905, 4921, 5045, 5104, 5164, 5217, 5328
\tex_noindent:D . . . . . 5353
\tex_par:D . . . . . 5359
\tex_pdfextension:D 5081, 5088, 5305
\tex_the:D . . . . . 1301, 4726, 5240
\tex_Umathaxis:D . . . . . 5375
\tex_vrule:D . . . . . 4805, 4850, 4869, 4905, 4921
\textbf . . . . . 127, 236, 239
\textcolor . . . . . 19, 40, 240
\textfont . . . . . 1232
\textnormal . . . . . 236
\textstyle . . . . . 5375
\textsubscript . . . 16, 36, 118, 123, 239
\textsuperscript . 16, 36, 118, 123, 239
tile . . . . . 4548
tl commands:
\tl_clear:N 254, 255, 300, 312, 574, 622, 959, 961, 963, 989, 998, 1038, 1163, 1262, 1369, 1371, 1457, 5015
\tl_const:Nn . . . . . 11
\tl_gclear:N . . . . . 425, 426
\tl_gset:Nn . . . . . 432, 433, 436, 437, 467, 469, 634, 636, 656, 658, 1510, 1522, 1620
\tl_gset_eq:NN 414, 418, 1103, 5289
\tl_if_blank:nTF . . . . . 933, 1366
\tl_if_empty:NTF . . . . . 227, 265, 635, 657, 935, 945, 968, 1010, 1039, 1122, 1168, 1212, 1375, 1442, 1448, 1547, 4720, 4820, 4837, 4845, 4994, 5155, 5237, 5266, 5284, 5296, 5320
\tl_if_empty_p:N . . . . . 4714
\tl_if_eq:NNTF . . . 990, 1051, 1617
\tl_if_eq:NnTF . . . . . 1120
\tl_if_eq:nnTF . . 813, 816, 830, 871
\tl_if_novalue:nTF . . . . . 1415, 1417, 1425, 1427, 4702
\tl_new:N . . . . . 94, 113, 114, 219, 220, 221, 224, 284, 286, 287, 288, 441, 443, 445, 447, 449, 459, 463, 464, 465, 466, 468, 473, 474, 478, 479, 480, 481, 482, 485, 486, 888, 890, 891, 892, 893, 895, 902, 904, 1136, 1137, 1206, 1256, 1344, 1348, 1350, 1509, 4657, 4658, 4660, 4661, 4662, 4671, 4672, 4673, 4674, 4675, 4676, 4681, 4683, 5208, 5340, 5580, 5604
\tl_put_left:Nn . . . . . 4775
\tl_put_right:Nn . . . . . 83, 229, 304, 307, 934, 941, 951, 1004, 1005, 1263, 1276, 4991
\tl_replace_all:Nnn . . . . . 292

\tl_set:Nn ..... 95, 104, 107, 120, 130, 228, 233, 236, 266, 278, 291, 294, 442, 444, 446, 448, 450, 460, 505, 507, 511, 514, 520, 522, 526, 528, 530, 532, 575, 594, 596, 611, 613, 621, 623, 889, 903, 970, 984, 1002, 1034, 1035, 1040, 1042, 1050, 1067, 1083, 1089, 1095, 1099, 1311, 1345, 1349, 1353, 1363, 1377, 1408, 1414, 1443, 1450, 1466, 1472, 1482, 1489, 4716, 4730, 4747, 4748, 4750, 4751, 4788, 4799, 4830, 4843, 4897, 4903, 4912, 4919, 4954, 4955, 5003, 5043, 5073, 5102, 5140, 5162, 5187, 5215, 5224, 5239, 5244, 5248, 5258, 5311, 5347, 5581, 5600, 5614, 5625
\tl_set_eq:NN ..... 932, 1123, 1124, 1209, 1213, 1351, 5285, 5291
\tl_to_str:n ..... 5146, 5225
\tl_trim_spaces:N ..... 234, 293
\tl_trim_spaces:n ..... 917
\tl_use:N ..... 12
token commands:
\token_to_str:N ... 154, 733, 1111
transparent commands:
transparent_node ..... 2416
\ttdefault ..... 1626
\typeout ..... 5498

U

u16 ..... 3298
u32 ..... 3298
\underLine ..... 4815
use commands:
\use:N ..... 1229, 4726, 4812, 4885, 5240, 5280, 5371
\use_i:nn ..... 1344, 1345
\use_ii:nn ..... 1346
\usepackage ..... 4, 23

V

vbox commands:
\vbox_set:Nw ..... 5350
\vbox_set_end: ..... 5361
\vbox_set_top:Nn ..... 5366
\vbox_unpack:N ..... 5367
\verb ..... 234, 240
VLIST ..... 1656

W

warn commands:
warn_disabled_cjk_math .... 3221
WHATSIT ..... 1656

Y

\yahei ..... 5518
\youyuan ..... 6, 26, 236, 1496, 5518

# Change History

v0.0.1

General: Provide LuaCJK-listings, which keeps CJK characters in place inside a listings block and counts each of them as two columns; LuaCJK loads it automatically with listings. . . . 1

Provide the CJK font family interface (\setCJKmainfont, \setCJKsansfont, \setCJKmonofont, \setCJKfamilyfont, \CJKfamily, \newCJKfontfamily, \CJKfontspec); an undeclared family warns once. . . . . 1

Refuse to load with luatexja. . . . 1

apply_check_single: Provide CheckSingle and WidowPenalty, which stop a single CJK character from ending up alone on the last line of a paragraph. . . . . 161

apply_kinsoku: Provide PunctKinsoku line-breaking prohibitions: closing punctuation cannot start a line and opening punctuation cannot end a line. . . . . 132

bound_width: Provide PunctBoundWidth, which leaves only the given advance of a mark inside the line and hangs the rest into the margin, at the end of a line, at its start, and when the mark ends a box such as \mbox. . . . . 144

CJKfilltwosides: Provide the CJKfilltwosides environment, which justifies every line to a given width and places the box on the baseline as requested. It is tagged: a text block in vertical mode, part of the surrounding paragraph in horizontal mode. 204

\CJKunderanyline: Provide \CJKunderanyline and \CJKunderanysymbol for arbitrary rules and repeated symbols; \CJKunderanysymbol puts one symbol per em, centered in the cell, its depth measured from the baseline to the top of the symbol. . . . . 199

\CJKunderdblline: Provide \CJKunderdblline, \CJKunderwave, and \CJKxout; the wave and the slash stay in phase along a line, their patterns scale with the font size, a decoration does not open up its line, and it fills its span exactly. . . . . 193

copy_glyph: A character the font lacks leaves no placeholder: the .notdef or private-use glyph a font substitutes for it is removed. . . . . 116

\defaultCJKfontfeatures: Provide \defaultCJKfontfeatures and \addCJKfontfeatures. . . . . 85

handle_boundary: A mark keeps its break opportunity beside suspended text: the boundary glue at a mark has no width. . 123

inherit_attributes: Glue, kerns, and penalties inserted by the package carry the attributes of the neighbouring node, so a decoration such as \CJKunderdot covers CJK/CJK

and CJK/Latin boundaries without a gap. . . . . 121
LuaCJK.cj_loaded: Provide CJLineBreak, which forbids a line break before the characters whose Unicode Line_Break property is CJ. . . . . 99
LuaCJK.define_punct_style: Provide KaiMingPunct, MiddlePunct, LongPunct, and NoBreakLongPunct, each with + and - forms, to edit the punctuation character sets. . . 108
__LuaCJK_add_fallback:nn: Provide \setCJKfallbackfamilyfont and AutoFallBack for characters that the selected CJK font does not have. . . . . 88
__LuaCJK_load_fandol:: Provide the LoadFandol option, which supplies Fandol fonts when the document declares none. . . . . 92
__LuaCJK_set_autofake:nn: Provide AutoFakeBold, AutoFakeSlant, EmboldenFactor, and SlantFactor; a family that sets BoldFont or ItalicFont is not emboldened or slanted. . . . . 57
__LuaCJK_set_punct_style:n: Provide the PunctStyle key with the quanjiao, plain, banjiao, kaiming, CCT, and hangmobanjiao styles. Under a style with a fixed width the advance of an opening or middle mark moves its ink with it. . . . . 48
__LuaCJK_settings_push:: Every setting travels with the characters: a snapshot of the options is recorded on the glyphs, so a group that ends before its paragraph decides how its own text is set. . . . . 64
__LuaCJK_setup_format_chars:: Ignore the zero-width format characters (U+200B, U+200C, U+200D, U+2060, U+180E) and the variation selectors at the input, so that they neither take glue nor separate two CJK characters. . . . . 90
__LuaCJK_update_font:: Provide PunctFamily, which sets a separate CJK family for the punctuation marks. . . . . 78
__LuaCJK_verbatim_env_end:: Suspend the adjustments inside \verb and the verbatim-like environments (verbatim, fancyvrb, listings); the Verb option (true, env, env+, false) controls what is suspended there. . . . . 69
\LuaCJKDeclareCharClass: Provide \LuaCJKDeclareCharClass, \LuaCJKDeclarePunctClass, and \LuaCJKResetCharClass to override the class of a code-point range. The classes carry the names of xeCJK (FullLeft, FullRight, HalfLeft, HalfRight, NormalSpace, CJStarter, CM, Boundary), the reset also restores the normal-spaced set, and the built-in table covers the CJK blocks of Unicode 15 and 16. . . . . 54
\LuaCJKDeclarePunctStyle: Provide \LuaCJKDeclarePunctStyle for a punctuation style of your own, with the keys compress-adjacent, adjacent-ratio, and keep-open-close. . . . . 56

\LuaCJKDeclareSubCJKBlock: Provide \LuaCJKDeclareSubCJKBlock with \LuaCJKCancelSubCJKBlock and \LuaCJKRestoreSubCJKBlock, so a range of characters can use its own font inside a family. . . . 51
\LuaCJKEditPunctStyle: Provide \LuaCJKEditPunctStyle to edit the punctuation sets of the current style, and AllowBreakBetweenPuncts. . . 49
\LuaCJKemdescent: Provide \LuaCJKemdescent, the depth of the em box of the current font, so that an object of height 1em placed at that depth covers the one-em square of a CJK character in any font. . . . . 199
\LuaCJKenable: Provide \LuaCJKdisable and \LuaCJKenable to suspend the typographic adjustments while keeping CJK font selection; the state is recorded per node. . . . . 64
\LuaCJKnobreak: Provide \LuaCJKnobreak and the PunctWidth option; neither this prohibition nor that of \LuaCJKVerbAddon depends on PunctKinsoku, which controls only the punctuation rules. . . . 65
\LuaCJKOffVerbAddon: Provide \LuaCJKVerbAddon and \LuaCJKOffVerbAddon, which make one CJK character as wide as two spaces of the current monospaced font and forbid breaks between CJK characters, so columns in code listings stay aligned. . . . . 65
\LuaCJKsetwidth: Provide \LuaCJKsetwidth and \LuaCJKsetkern for per-mark widths and per-pair gaps that override the selected punctuation style. . . . . 50
\LuaCJKShipoutHook: Provide \LuaCJKShipoutHook, which runs code through the standard shipout/before hook. . . . . 65
\LuaCJKversion: Provide \LuaCJKversion, which prints the version of the package. . . . 45
\normalspacedchars: Provide \normalspacedchars so that slashes, backslashes, and hyphens take no glue on either side. . . . . 90
rewrite_fonts: Provide LatinPunct, which keeps the shared quotation marks, the middle dot, and the ellipses in the Latin font instead of the CJK font. . . . . 117
rewrite_math: Provide CJKmath and \setCJKmathfont for CJK characters in math mode, including nested math lists such as fractions. The family is registered as a math family, so a character in a script or a fraction keeps the size of that math style; with CJKmath off, a CJK character in math mode warns once. . . . . 140
spacing: Provide CJK/CJK and CJK/Latin spacing with CJKglue, CJKecglue, CJKspace, and xCJKecglue; a value keeps its stretch and shrink orders. A source space between two CJK characters becomes the inter-character glue, so the boundary keeps its break opportunity and its stretch, and no glue is inserted between two long punctuation marks. . . . . 123

\youyuan: Provide \songti, \heiti, \fangsong, \kaishu, \lishu, and \youyuan as shorthands for the families zhsong, zhhei, zhfs, zhkai, zhli, and zhyou. . . . . 89
v0.0.2
apply_kinsoku: AllowBreakBetweenPuncts no longer lets a closing mark start a line or an opening mark end one, and a mark inside a box takes part in the prohibitions. 132
\CJKfontspec: \CJKfontspec and \newCJKfontfamily take their options after the font name as well. . . . . 89
\CJKunderdblline: The double rule takes format like the other decorations, and a rule or a symbol that carries no format follows the color of the surrounding text. . . . . 193
hang_bounds: The hanging of a mark takes part in line breaking, so a line that ends at a closing mark or begins with an opening mark holds the mark the margin allows; the ink of that mark stands outside the line, and PunctBoundWidth decides how much of the advance stays inside. . . . . 158
__LuaCJK_add_fallback:nn: The fallback list is resolved again at every font selection, so the fallback follows the series, shape, and size of the text. . . . . 88
__LuaCJK_parse_font_options:n: Baseline shifts the glyphs of the font that declares it, so two CJK fonts whose baselines do not match can be set side by side. . . . . 52
__LuaCJK_reset_family:: Restore \CJKfamilydefault when \normalfont or \textnormal runs, so that the default font command overrides a family chosen earlier in the group; \CJKfamilydefault follows \CJKrmdefault until the document sets it. . . . . 76
\LuaCJKnobreak: \LuaCJKnobreak also forbids the break at a discretionary, so an explicit hyphen or an automatic hyphenation point inside the argument no longer breaks the line. . . . . 65
mark_blank: The hang of a mark is the blank its font carries, read from the outline, so the ink of a line-final or line-initial mark stands on the measure instead of half an em inside it; a file that cannot be read falls back to half the advance. . . . . 155
restore_source_spaces: The space in front of \textbf and its siblings is still the inter-character glue: LaTeX moves a space across the italic correction of a text font command, and the move leaves the glue with the subtype of a skip the document wrote. . . . 127
\setCJKmathfont: A CJK character in a formula takes the face of the math version: the bold face of the family under \boldmath when the family has one, and the upright shape instead of the shape of the text. . . . . 89
v0.0.3
General: A mark that the CJK class shares with Latin, such as a curly quotation mark or the em dash, keeps its place inside an lstlisting block and counts as

two columns: the character table covers the CJK ranges of the Basic Multilingual Plane, both ends of each included. . . . 1
The punctuation the two scripts share is a CJK character: the middle dot (U+00B7), the horizontal bar (U+2015), the two dot leader (U+2025), the hyphenation point (U+2027), and the midline ellipsis (U+22EF) are set in the CJK font, and LatinPunct keeps the ones it lists in the Latin font; the two dot leader joins a run of dashes and ellipses that may not be broken. . . . . 1
apply_check_single: A whatsit between two CJK characters no longer hides the break point from CheckSingle: a color change, a mark, and the other nodes that carry no text are passed over, so they no longer leave a single character on the last line. . . . . 161
apply_kinsoku: A dash or an ellipsis cannot start a line either: a break in front of a run of them is forbidden, so the run travels to the next line whole, while the break between the marks of a run stays the business of the NoBreakLongPunct set. . . . . 132
box_visible: A box that occupies no advance is passed over: \rlap, \llap and \clap overlay ink on the text around them, so they no longer take a CJKecglue on either side. . . . . 118
cff_box: A curve of four operands is read as one curve in both axes, so the blank of a mark whose charstring draws it with the alternating or the vertical curve operator is the blank the glyph has: a double opening quotation mark no longer stands in the margin by more than a tenth of a point. . . . . 151
The blank of a mark is measured from the charstring the glyph uses, so a glyph whose charstring calls a subroutine is measured in the subroutine rather than in the charstring the measurement began with. . . . 151
\CJKfamily: \CJKfamily expands its argument, so a family can be given as \CJKrmdefault and friends. . . . . 88
CJKfilltwosides: The t and c positions of CJKfilltwosides place the box as \parbox does: t puts the baseline of its first line on the surrounding baseline, and c centres the box on the text axis instead of on the baseline. 204
\CJKunderdot: The emphasis dots sit under each CJK character, centred in the one-em cell of the character; Latin letters, digits, and spaces carry none, and the dots follow the characters when the inter-character glue stretches. . . . . 203
\CJKunderline: hidden takes the ink out of the text instead of putting it in a \phantom: the text keeps its line breaks, so a hidden paragraph breaks like the visible one instead of overflowing in one line. . . . . 184
A decoration keeps the width, the stretch, and the shrink of the text it covers, so a rule no longer changes the spacing of the line it sits in, skip no longer tightens a justified line, and a \kern inside the text no longer moves

the gap skip leaves at a punctuation mark. . . . . 184
\CJKunderwave: The wave is a smooth arch of one period per em, a quarter em above and below the underline; a decoration is divided into whole patterns once its final width is known and the pattern is drawn at that width, so the periods meet without a gap, the wave reaches the last character, and its period follows the stretch of the line. . . . . 194
The wave takes format like the other decorations: the pattern is stroked in that color, in the color of its text when the key is absent, and neither color reaches the text outside it. . . . 194
\CJKxout: The slash is divided into whole cells once its width is final, so the slashes cover the span evenly and the last one reaches its end. . . . . 198
The slash takes format like the wave, and follows the color of its text when the key is absent. 198
drop_orphans: A joiner or a variation selector that the font cannot use leaves nothing behind, and neither does a CJK character the CJK font does not have. . . . . 115
handle_boundary: A penalty that prohibits the break no longer removes the glue: the CJK–CJK and CJK–Latin spacing survives \nobreak, because the glue goes after the penalty, where the prohibition still covers it. A source space in front of the penalty is converted and the penalty moves in front of it. . . 123
hang_bounds: An opening mark that begins a paragraph, or a line the document forces a break at, hangs as well: its blank stands in the indentation or in the margin, and PunctBoundWidth decides how much of the advance stays inside the line there too. . . . . 158
__LuaCJK_ensure_family_default:: A document whose default CJK family is not declared is reported, and its text takes a family it did declare instead of none at all. . . . . 76
__LuaCJK_fv_patch:: The Verb option also reaches the inline \Verb and \UseVerb of fancyvrb, which never select \verbatim@font and are set as \verb sets them; the block commands \VerbatimInput and \UseVerbatim follow the environments instead. . . . . 68
__LuaCJK_load_fandol:: LoadFandol gives the roman family FandolKai-Regular.otf as its italic face, so \emph and \itshape set the text in Kai instead of leaving it upright. . . 92
LoadFandol is on by default, so a document that declares no CJK font is set in the four Fandol faces without asking for them. . 92
__LuaCJK_math_fallbacks_push:n: A CJK character in math mode takes a fallback font when the CJK font of the formula does not have it. . . . . 82
__LuaCJK_register_math_family:n: The three sizes of a CJK math family come from the text size of the formula, so a \text inside it no longer sets the other characters of the formula in a script size. . . . . 81

__LuaCJK_setup_format_chars:: The joiners (U+200C, U+200D) and the variation selectors are no longer removed at the input: they reach the font, so an emoji sequence is not split and a CJK variation sequence selects its glyph, and what the font leaves of them is dropped before the spacing pass, so two CJK characters still take no glue around them. . . . . 90
__LuaCJK_update_font:: PunctFamily shifts its marks by the Baseline of the family it names. . . . . 78
\LuaCJKOffVerbAddon: true and env as well; \LuaCJKOffVerbAddon restores the suspension that was in force. . . . . 65
\LuaCJKsetup: PunctWidth and PunctBoundWidth take 0pt as a length: the former gives every mark no advance and the latter hangs the whole of it, where a zero length used to act as false and return the mark to its style. 58
Process the package options after the helpers their key codes call, so that AutoFallBack, CJKmath, PunctFamily, and CJLineBreak act as package options as well. . 58
pre_shaping: The CJK font is applied to the characters before the text is shaped instead of to the glyph nodes after it: a combining mark attaches to its base, a variation sequence selects its glyph, and the font's own ligatures and locl forms appear. An older luaotfload keeps the previous behavior. . 114
restore_source_spaces: \hfill after \textbf keeps its fill: only a glue of normal order counts as a source space, so the fill of the line is no longer turned into the inter-character glue. . . . . 127
\setCJKfamilyfont: \setCJKfamilyfont expands the name of the family it declares, so a family can be given as \CJKrmdefault and friends. . . . . 86
spacing: A superscript or subscript set with \textsuperscript or \textsubscript stays with the character in front of it: the shifted box takes no glue on either side, so no break is offered before it. . . . . 123
v0.0.4
General: A run of CJK characters inside an lstlisting block is one token of listings, so a emph or keywords entry that names a CJK word highlights it; a Latin character and a CJK mark both end the run. . . . . 1
apply_fixed_widths: A punctuation mark is given its fixed advance once: a box that holds the list is packed again, and the mark no longer takes a second compensating kern, so a fixed-width style no longer squeezes a table cell or a parbox. . . . . 137
\CJKfamilydefault: An untouched \CJKfamilydefault follows \familydefault at the start of the document, so a preamble that redefines \familydefault sets the main text in the matching CJK family. 85
The family \CJKfamilydefault names is selected when the document starts, so a preamble that redefines it sets the main

text in that family from the first line. . . . . 85
CJKfilltwosides: CJKfilltwosides leaves vertical mode as \parbox does, so the text after it joins the box on the same line instead of beginning a paragraph, and a space after the environment is ignored as xeCJK ignores it. . 204
\CJKfontspec: \CJKfontspec no longer reads an option list after the font name, so a bracket that the text starts with is no longer swallowed. . . . . 89
\CJKunderdblline: format is resolved at every call, so a color that \colorlet redefines between two decorations is followed by the second one instead of keeping the color of the first. . . . . 193
\CJKunderdot: A CJK character inside a box gets its dot wherever the box sits in the span, so \mbox no longer drops the dot of a character in the middle of the text. . . . . 203
\CJKunderline: hidden hides a formula as it hides text: a character or a rule inside a box of the span carries no ink, so a fraction is hidden like a plain number and keeps its width, height, and depth. . . . . 184
\CJKunderwave: The wave is written one period at a time, so it covers the start and the end of a span that is not a whole number of periods, on a stretched line as well as on the last line of a paragraph. . . . . 194
\CJKxout: The slashes are written one cell at a time, so they cover the whole span. . . . . 198
\defaultCJKfontfeatures: \defaultCJKfontfeatures reaches the font of a sub-block and of a fallback font as well. . . 85
handle_boundary: A space in front of a color change keeps its advance: the whatsit a \textcolor inserts no longer hides the space from the boundary rules, so no CJKecglue or second CJKglue joins it. . . . . 123
A suspended run keeps the break opportunity beside it: the zero-width glue at its edge lets \verb end a line next to a CJK character while the spacing stays removed. . . . . 123
hang_bounds: A centered or flush-right paragraph does not hang the mark it begins with: the line does not start at the edge of the measure, so the shift would move the whole line instead of leaving the blank in a margin that is not there. . . . 158
The first line of a list item hangs as well: the label box that the list writes in front of the item's text is passed over, so a mark that begins the item leaves its blank in the label's advance. . 158
LuaCJK.hyphenate: A CJK character ends a word for hyphenation, so the line breaker no longer puts a hyphen beside a CJK character, and the boundary keeps the CJKecglue that the two sides give it. . . . 164
LuaCJK.is_cjk: A code point below U+0100 in a font that is not Unicode-encoded is a position in that font's encoding, not a character, so a symbol font's slot is no longer read as a

CJK character: pifont's slot 0xB7, the circled 2, keeps its own font and prints. . . . . 100
With LatinPunct on, the midline ellipsis U+22EF stays in the Latin font like the other shared marks that key lists. . . . . 100
__LuaCJK_family_font:n: The font of a CJK family is resolved in the Unicode encoding: a document that sets its Latin text in an 8-bit encoding with fontenc keeps its CJK text. . . . . 70
__LuaCJK_parse_font_options:n: Baseline belongs to the declaration that writes it, so a fallback font and a family that name the same font keep their own offsets. . . . . 52
\LuaCJKDeclareCharClass: HalfLeft and HalfRight declare a half-width ASCII bracket: it keeps its advance, does not hang, and hugs the text inside the pair, where they used to declare a full-width opening or closing mark. . . . . 54
\LuaCJKnobreak: \LuaCJKnobreak forbids the break at a glue the document writes inside its argument, so a \quad or an \hspace there no longer starts a line. . . . . 65
PUNCT_CLASS: The ASCII brackets are punctuation marks of their own: a half-width opening bracket no longer ends a line and a half-width closing bracket no longer starts one, the pair hugs the text between it, and a compression style leaves the bracket's own advance alone. 103
The punctuation table covers the marks xeCJK classes as opening or closing, so the percent sign U+FF05 no longer starts a line and the currency symbols no longer end one; the halfwidth and compatibility forms and the three hyphen-like marks follow the same class. . . . . 103