LuaCJK 宏包
The LuaCJK package*

Mingyu Xia†

Released 2026-11-15 v0.0.2

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，并在断行与盒子打包之前替换 CJK 字形节点。

LuaCJK provides Chinese typesetting for LuaTEX. It records the current CJK font in a LuaTEX attribute and replaces CJK glyph nodes before line breaking and box packing.

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 8
3.6 数学模式中的汉字 9

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

4 文本排版 10
4.1 间距 . . . 10
4.2 标点 . . . 11
4.3 断行 . . . 14
4.4 暂停与钩子 . . . 15
4.5 抄录与 listings . . . 16
4.6 装饰 . . . 16

5 与 xeCJK 的比较 19
5.1 行为差异 . . . 20
5.2 未提供的功能 . . . 20

1 Introduction 21

2 Loading 21

3 Fonts 23
3.1 Sub-blocks . . . 24
3.2 Fallback fonts . . . 25
3.3 Baseline shift . . . 25
3.4 Loading Fandol . . . 26
3.5 Fake bold and slant . . . 26
3.6 CJK in mathematics . . . 27

4 Text layout 28
4.1 Spacing . . . 28
4.2 Punctuation . . . 29
4.3 Line breaking . . . 33
4.4 Suspending and hooks . . . 34
4.5 Verbatim and listings . . . 34
4.6 Decorations . . . 35

5 Comparison with xeCJK 39
5.1 Different behavior . . . 39
5.2 Not available . . . 39

A Implementation **41**
A.1 Loading and conflicts 41
A.2 Registers, styles, and options 42
A.3 Fonts 63
A.4 Names and configuration 78
A.5 The Lua module 81
A.6 Settings and data 83
A.7 Node rewriting 94
A.8 The passes 103
A.9 Decorations 137
A.10 Listings and ctex 163

Index **169**

Change History **185**

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，并在断行与盒子打包之前替换 CJK 字形节点。

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；
- LoadFandol 与 LocalConfig 在加载宏包时起作用。

声　明　　\LuaCJKDeclareCharClass、\LuaCJKDeclarePunctClass、\LuaCJKDeclareSubCJKBlock、\LuaCJKsetwidth 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。

\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。

\CJKfamily `\CJKfamily {⟨family⟩}`

切换到具名的 CJK 字族。

```
\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⟩} [⟨options⟩]`

为当前分组选择一个 CJK 字体。选项可以写在字体名之前，也可以写在之后。

\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 \CJKrmdefault。

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。

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，共享后一条声明的值，因为字的位置是按字体记录的。

```
\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*⟩ 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。

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。

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；一个文档如果在数学里用到十六个以上的 CJK 字族，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。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 1filll）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。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。

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
\normalspacedchars{x}中x中
```

中 x 中
中x中

只用于控制文字、本身没有进距的字符（U+200B、U+200C、U+200D、U+2060、U+180E、U+FEFF 以及各个变体选择符）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：零宽空格不会加上本来会在它两侧出现的两个 CJKecglue，而它想提供的断行机会在周围的胶上本来就存在。用 \char 请求的字符不会被删除，因为 \char 要的就是字形本身。

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*⟩ 参数决定盒子在基线上的位置： t 表示顶部，c 表示居中（默认），b 表示底部。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。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。 `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）与省略号（U+2026、U+2027、U+22EF）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。

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。收尾的或中间的标点不能出现在行首、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。收尾的或中间的标点不能出现在行首，开头的标点不能出现在行尾，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！

```
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；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；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。键 LongPunct 替换这个集合。

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。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。 \LuaCJKOffVerbAddon 在同一个分组里再次取消它。

```
\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：紧挨着被暂停材料的一个标点，仍然保留它后面的断行机会。

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。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 被传给 lua-ul，因此只对 \CJKunderline 有效，并覆盖计算出来的几何。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。

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。

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。

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，并用 `\leaders` 重复，所以在一行的各部分之间相位相同。一个周期是半字宽，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。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 字体是在文字用当前字体定形之后才应用到字形节点上的，所以 CJK 字体的 OpenType 特性（连字、`locl`、`palt`、变体序列）不生效：间距、标点进距与禁则都来自 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`，理由同上：字体的两字宽破折号连字在 LuaCJK 应用 CJK 字体之前就已经定了。
- `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 CJK glyph nodes are replaced before line breaking and box packing.

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;
- `LoadFandol` and `LocalConfig` act while the package is loaded.

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.

\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.

\CJKfamily

`\CJKfamily {⟨family⟩}`

Switch to a named CJK family.

```
\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⟩} [⟨options⟩]`

Select a CJK font for the current group. The options may be given before or after the font name.

\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 \CJKrmdefault by default.

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.

`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 share the value of the last one, because a position is recorded per font.

```
\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*⟩ When the document does not declare any 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.

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.

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

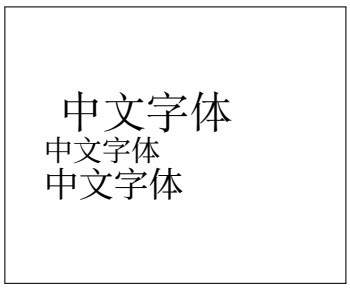
\defaultCJKfontfeatures `\defaultCJKfontfeatures {⟨font features⟩}`

Set the features prepended to every CJK family registered afterwards. 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
```



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_ε allocates under LuaTeX, which an ordinary document never reaches; more than sixteen CJK families in math stops with an error. The script sizes are seven and five tenths of the current text size, the ratio of the standard size tables. 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. 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.

`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
\normalspacedchars{x}中x中
```

中x中
中x中

Characters that only control the text and have no advance of their own (U+200B, U+200C, U+200D, U+2060, U+180E, U+FEFF, and the variation selectors) are removed at the input. They therefore neither take glue nor separate two CJK characters: a zero-width space does not add the two `CJKecglue` that would otherwise surround it, and the break opportunity it was meant to create already exists at the surrounding glue. A character requested with `\char` is not removed, because `\char` asks for the glyph itself.

`CJKfilltwosides` (*env.*) Set the lines in a vertical box whose lines are justified to ⟨*width*⟩, separated by `\\`. Its ⟨*position*⟩ argument puts the box on the baseline: `t` for top, `c` for centered (the default), or `b` for bottom. 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，\\虛室生白
\end{CJKfilltwosides}
```

瞻彼闋者，
虛室生白

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

瞻 彼 闋 者 ，
虛 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. 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. 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. `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), and the ellipses (U+2026, U+2027, U+22EF) 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.

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 or middle 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 or middle punctuation cannot start a line, an opening punctuation cannot end a line, 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！

```
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; `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`; `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. The key LongPunct replaces the set.

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. 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. `\LuaCJKOffVerbAddon` cancels it again inside the same group.

```
\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: a punctuation mark next to suspended material still keeps the break opportunity that follows it.

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. 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` are passed to lua-ul, so they apply to `\CJKunderline` alone and override the computed geometry. A drawn pattern never changes the height or depth of the line it decorates, so the leading does not grow.

`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`.

`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.

`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 repeated with `\leaders`, so it has the same phase across the parts of one line. One period is half an em and the whole pattern scales with the font size; 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
• • • • •

\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*⟩. 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 glyph node after the text has been shaped with the current font, so the OpenType features of the CJK font (ligatures, `locl`, `palt`, variation sequences) do not take effect: the spacing, the punctuation advance, and the prohibitions 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`, for the reason given above: the font's two-em dash ligature is decided before LuaCJK applies the CJK font.
- `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 21.*)

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

The attributes record, per node, the CJK font, the space and glue policies, whether the node was inserted by LuaCJK, the CJK math font, and whether the adjustments are suspended. 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
  }
  {
    \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`.)

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* 32.)

`\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 32.*)

`\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 32.*)

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 24.*)

`\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* 25.)

`\__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:
          \fontspec_gset_family:Nnn \l__LuaCJK_tmpa_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 { "open"  } {#2} }
        { FullRight   } { \__LuaCJK_push_punct:nn { "close" } {#2} }
        { HalfRight   } { \__LuaCJK_push_punct:nn { "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 31.*)

`\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} }
        { 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 31.*)

`\__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 32.*)

`\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 32.*)

`\__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 switches read by the Lua side, the `Verb` setting, 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_break_between_puncts_bool
\dim_new:N  \l__LuaCJK_punct_width_dim
\dim_set:Nn \l__LuaCJK_punct_width_dim { 0pt }
\dim_new:N  \l__LuaCJK_punct_bound_width_dim
\dim_set:Nn \l__LuaCJK_punct_bound_width_dim { 0pt }
\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_family_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 }
        { \dim_zero:N \l__LuaCJK_punct_width_dim      }
        { \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 }
        { \dim_zero:N \l__LuaCJK_punct_bound_width_dim }
        { \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 } },
```

```
572     CheckSingle         .default:n   = { true },
573     WidowPenalty        .code:n      =
574       { \int_set:Nn \l__LuaCJK_widow_penalty_int {#1} },
575     LocalConfig         .code:n      =
576       {
577         \str_case:nnF {#1}
578           {
579             { true  } { \tl_set:Nn \l__LuaCJK_local_config_tl { luacjk.cfg } }
580             { false } { \tl_clear:N \l__LuaCJK_local_config_tl           }
581           } { \tl_set:Nn \l__LuaCJK_local_config_tl { luacjk-
#1.cfg } }
582       },
583     LocalConfig         .default:n   = { true  },
584     AutoFakeBold        .code:n      = { \__LuaCJK_set_autofake:nn { bold  } {#1} },
585     AutoFakeBold        .default:n   = { true  },
586     AutoFakeBold        .initial:n   = { false },
587     AutoFakeSlant       .code:n      = { \__LuaCJK_set_autofake:nn { slant } {#1} },
588     AutoFakeSlant       .default:n   = { true  },
589     AutoFakeSlant       .initial:n   = { false },
590     EmboldenFactor      .code:n      =
591       {
592         \tl_gset:Nn \g__LuaCJK_embolden_factor_tl {#1}
593         \tl_if_empty:NF \g__LuaCJK_autofake_bold_tl
594           { \tl_gset:Nn \g__LuaCJK_autofake_bold_tl {#1} }
595       },
596     AutoFallBack        .choice:,
597     AutoFallBack / true .code:n      =
598       {
599         \bool_set_true:N \l__LuaCJK_auto_fallback_bool
600         \__LuaCJK_refresh_fallbacks:
601       },
602     AutoFallBack / false.code:n      =
603       { \bool_set_false:N \l__LuaCJK_auto_fallback_bool },
604     AutoFallBack        .default:n   = { true  },
605     LoadFandol          .choice:,
606     LoadFandol / true   .code:n      =
607       { \bool_gset_true:N \g__LuaCJK_load_fandol_bool },
608     LoadFandol / false  .code:n      =
609       { \bool_gset_false:N \g__LuaCJK_load_fandol_bool },
610     LoadFandol          .default:n   = { true  },
611     SlantFactor         .code:n      =
612       {
613         \tl_gset:Nn \g__LuaCJK_slant_factor_tl {#1}
614         \tl_if_empty:NF \g__LuaCJK_autofake_slant_tl
615           { \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:
  }
\ProcessKeyOptions [ LuaCJK / options ]
```

(*End of definition for* `\LuaCJKsetup` *and others. This function is documented on page 21.*)

`\__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_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 ,
                \dim_to_decimal_in_sp:n { \l__LuaCJK_punct_width_dim } ,
                \dim_to_decimal_in_sp:n { \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* 34.)

`\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* 34.)

`\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.

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

(*End of definition for* `\LuaCJKnobreak`. *This function is documented on page* 33.)

`\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 width of two spaces, the width of one CJK character, the gap between them, the saved glues, whether the addon is 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
\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 \fontfamily \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
        \__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
        \bool_set_false:N \l__LuaCJK_verb_addon_bool
      }
```

```
  }
```

(*End of definition for* `\LuaCJKVerbAddon` *and others. These functions are documented on page 34.*)

`\__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_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.

```
\cs_new_protected:Npn \__LuaCJK_verbatim_env_begin:
  {
    \tl_if_eq:VnT \l__LuaCJK_verb_mode_tl { true }
      { \LuaCJKdisable }
  }
\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_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`
`__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}
  }
```

__LuaCJK_build_features:n

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

```
\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
        {
          \prop_gput:NVV \g__LuaCJK_font_shift_prop
            \l__LuaCJK_fontspec_tl \l__LuaCJK_shift_dim_tl
        }
    \group_end:
    \prop_if_in:NVT \g__LuaCJK_family_prop \l__LuaCJK_family_tl
      { \selectfont }
  }
```

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.

```
\__LuaCJK_push_fallback:
_LuaCJK_refresh_fallbacks:
\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}
        {
          \fontfamily {##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:nn { \l__LuaCJK_tmpb_tl } {##1}
        }
      \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:
    \bool_set_false:N \l__LuaCJK_selectfont_bool
  }
```

`\__LuaCJK_push_shift:nn`

Hand the vertical offset of one font to Lua, once per font identifier. The `Baseline` value of each registered font, keyed by the family fontspec returned for it; the identifiers whose offset Lua already knows, and scratch variables for the identifier and the offset, which is evaluated at the size in force so that a value in `em` follows the text.

```
\cs_new_protected:Npn \__LuaCJK_push_shift:nn #1#2
  {
    \tl_set:Ne \l__LuaCJK_shift_fid_tl {#1}
    \tl_set:Ne \l__LuaCJK_shift_family_tl {#2}
    \prop_get:NVNT \g__LuaCJK_font_shift_prop
      \l__LuaCJK_shift_family_tl \l__LuaCJK_shift_dim_tl
      {
        \tl_if_empty:NF \l__LuaCJK_shift_dim_tl
          {
            \prop_if_in:NVF \g__LuaCJK_shift_pushed_prop \l__LuaCJK_shift_fid_tl
              {
                \prop_gput:NVn \g__LuaCJK_shift_pushed_prop
                  \l__LuaCJK_shift_fid_tl { }
                \lua_now:e
                  {
                    LuaCJK.set_yoffset
                      ( \l__LuaCJK_shift_fid_tl ,
                        \dim_to_decimal_in_sp:n
                          { \dim_eval:n { \l__LuaCJK_shift_dim_tl } } )
                  }
              }
          }
      }
  }
```

(*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 a document that has declared no font yet 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_math_font_update:` Register the CJK family used by the current settings as a math family, 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
```

```
994           { \exp_args:NV \__LuaCJK_register_math_family:n \l__LuaCJK_tmpb_tl }
995           { \g__LuaCJK_math_font_attr = \c_zero_int }
996       }
997       { \g__LuaCJK_math_font_attr = \c_zero_int }
998   }
```

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

`\__LuaCJK_update_font:` On every `\selectfont`, resolve the current family, its fallback families, and the math family to font identifiers and store them in the attributes. A guard against re-entering the update while `\selectfont` runs, and the font identifier of the current CJK family.

```
999 \bool_new:N \l__LuaCJK_selectfont_bool
1000 \int_new:N \g__LuaCJK_fontid_int
```

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

`\l__LuaCJK_punct_family_tl`
`\g__LuaCJK_punct_fontid_int`

```
1001 \tl_new:N \l__LuaCJK_math_family_tl
1002 \tl_new:N \l__LuaCJK_punct_family_tl
1003 \int_new:N \g__LuaCJK_punct_fontid_int
```

`\g__LuaCJK_math_family_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_ε allocates the 256 families that LuaTeX allows from zero upwards, so the numbers start at 240; a document with more than sixteen CJK families is out of luck and gets an error.

`\g__LuaCJK_math_family_prop`
`\c__LuaCJK_math_family_max_int`

```
1004 \int_new:N \g__LuaCJK_math_family_int
1005 \int_gset:Nn \g__LuaCJK_math_family_int { 239 }
1006 \int_const:Nn \c__LuaCJK_math_family_max_int { 255 }
1007 \prop_new:N \g__LuaCJK_math_family_prop

1008 \cs_new_protected:Npn \__LuaCJK_update_font:
1009   {
1010     \bool_if:NF \l__LuaCJK_selectfont_bool
1011       {
1012         \bool_set_true:N \l__LuaCJK_selectfont_bool
1013         \prop_get:NVNTF \g__LuaCJK_family_prop \l__LuaCJK_family_tl \l__LuaCJK_family_
1014           {
1015             \group_begin:
1016               \exp_args:NV \fontfamily \l__LuaCJK_family_name_tl
1017               \selectfont
1018               \int_gset:Nn \g__LuaCJK_fontid_int
1019                 { \tex_fontid:D \tex_font:D }
```

```
            \__LuaCJK_push_shift:nn
              { \int_use:N \g__LuaCJK_fontid_int } { \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 \fontfamily \l__LuaCJK_tmpa_tl
                \selectfont
                \int_gset:Nn \g__LuaCJK_punct_fontid_int
                  { \tex_fontid:D \tex_font:D }
              \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_register_math_family:n` Give a CJK family a math family number and fill the three style fonts of that number from the family at the current size. 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. The script sizes are seven and five tenths of the text size, 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.

```
\msg_new:nnn { LuaCJK } { math-family-exhausted }
  { Too ~ many ~ CJK ~ 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
      }
  }
\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} }
    \group_begin:
      \fontfamily {#1}
      \fontseries { \mddefault }
      \fontshape { \shapedefault }
      \fontsize { \use:c { f@size } 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 \l__LuaCJK_tmpa_int = \font
      \fontsize
        { \dim_eval:n { \use:c { f@size } pt * 7 / 10 } }
        { \baselineskip }
      \selectfont
      \global \scriptfont \l__LuaCJK_tmpa_int = \font
      \fontsize
        { \dim_eval:n { \use:c { f@size } pt * 5 / 10 } }
```

```
        { \baselineskip }
      \selectfont
      \global \scriptscriptfont \l__LuaCJK_tmpa_int = \font
    \group_end:
    \g__LuaCJK_math_font_attr = \l__LuaCJK_tmpa_int
  }
```

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

`\__LuaCJK_math_version_refresh:` Register the math family again when the math version changes. LaTeX 2_ε 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 \fontfamily \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 27.*)

\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
            {
              \prop_gput:NVV \g__LuaCJK_font_shift_prop
                \l__LuaCJK_fontspec_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 27.*)

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

The public font commands, all built on `\__LuaCJK_set_family_with_options:nnnn`.

```
\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 }
```

```
  { \__LuaCJK_set_family_with_options:nnnn {#1} {#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}
```

__LuaCJK_set_family_with_options:nnnn

```
      {
        \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}
  }
```

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

```
\cs_new_protected:Npn \__LuaCJK_add_fallback:nn #1#2
  {
```

__LuaCJK_add_fallback:nn

```
    \__LuaCJK_parse_font_options:n {#1}
    \tl_if_empty:NT \l__LuaCJK_shift_tl
      { \tl_set:Nn \l__LuaCJK_features_tl {#1} }
    \group_begin:
      \fontspec_gset_family:NVn
        \l__LuaCJK_fallback_family_tl \l__LuaCJK_features_tl {#2}
      \tl_if_empty:NF \l__LuaCJK_shift_tl
        {
          \prop_gput:NVV \g__LuaCJK_font_shift_prop
            \l__LuaCJK_fallback_family_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 23.*)

`\CJKfamily` Switch to a registered family and gobble the following spaces.

```
\NewDocumentCommand \CJKfamily { m }
  { \__LuaCJK_switch_family:n {#1} \tex_ignorespaces:D }
```

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

`\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 23.*)

`\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 27.*)

`\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 o }
  {
    \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} {#3}
    \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 23.*)

\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 24.*)

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 29.*)

`\__LuaCJK_ignore_format_chars:` These characters have no advance of their own, but the font loader replaces one that the font does not have by a space glyph and a negative kern. The spacing pass then sees a Latin atom and surrounds it with `CJKecglue` on both sides, so a zero-width space would widen the line by two `CJKecglue` and remove the break opportunity it was meant to create. Giving the characters category 9 removes them before the font loader runs. 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.

```
\clist_const:Nn \c__LuaCJK_ignored_chars_clist
  { "180E , "200B , "200C , "200D , "2060 , "FEFF }
\cs_new_protected:Npn \__LuaCJK_ignore_format_chars:
  {
    \clist_map_inline:Nn \c__LuaCJK_ignored_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_ignore_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.

```
\cs_new_protected:Npn \__LuaCJK_load_fandol:
  {
    \setCJKmainfont{FandolSong-Regular.otf}
      [ BoldFont = FandolSong-Bold.otf ]
    \setCJKsansfont{FandolHei-Regular.otf}
      [ BoldFont = FandolHei-Bold.otf ]
    \setCJKmonofont{FandolFang-Regular.otf}
    \setCJKfamilyfont{zhsong}{FandolSong-Regular.otf}
      [ BoldFont = FandolSong-Bold.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.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_settings_push:
\__LuaCJK_set_normal_spaced_chars:
\__LuaCJK_ignore_format_chars:
\hook_gput_code:nnn { begindocument } { LuaCJK }
  {
    \tl_if_eq:NNT \CJKfamilydefault \l__LuaCJK_family_default_init_tl
      { \tl_gset:Nn \CJKfamilydefault { \CJKrmdefault } }
    \bool_if:NT \g__LuaCJK_load_fandol_bool
      {
        \prop_if_empty:NT \g__LuaCJK_family_prop
          { \__LuaCJK_load_fandol: }
      }
    \__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.

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 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")
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.attr_font
LuaCJK.attr_cjk_space
LuaCJK.attr_xecglue
LuaCJK.attr_inserted
LuaCJK.attr_math_font

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

```
for _, field in ipairs {
```

```
  "font",      "cjk_space",  "xecglue", "inserted",
  "math_font", "punct_font", "nobreak", "settings",
} 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   = 0,
  bound_width   = 0,
  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),
    punct_width, 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` `LuaCJK.cj_count` `LuaCJK.cj_loaded` The characters whose `Line_Break` property is `CJ` (the Japanese small kana and the like), how many there are, and whether the file has been read. The set is read 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

function M.load_cj ()
  if M.cj_loaded then
    return M.cj_count
  end
  M.cj_loaded = true
  local path = kpse and kpse.find_file and kpse.find_file("LineBreak.txt")
  local file = path and io.open(path)
  if not file then
    texio.write_nl("term and log",
      "Package LuaCJK Warning: LineBreak.txt not found; "
      .. "CJLineBreak = strict has no effect.")
    return 0
  end
  local data = file:read("*a")
  file:close()
  M.cj, M.cj_count = read_cj(data)
  return M.cj_count
end

```

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

The quotation marks, the middle dot, and the ellipses are shared with Latin

scripts. With `LatinPunct` on, they 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 (U+2014, the quotation marks, U+2026) 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 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 = {
  { 0x1100,  0x11FF },  -- Hangul Jamo
  { 0x2014,  0x2014 },  -- em dash, shared with Latin
  { 0x2018,  0x2019 },  -- single quotation marks
  { 0x201C,  0x201D },  -- double quotation marks
  { 0x2026,  0x2026 },  -- horizontal 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,
}

local function is_cjk (c, s)
  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)
end

M.is_cjk = is_cjk

```

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

`PUNCT_CLASS` The punctuation LuaCJK knows about, grouped as opening, closing, middle, dash, and ellipsis. The group decides compression, line-breaking prohibitions, and hanging.

```
local PUNCT_CLASS = {
  [0x2018] = "open",     [0x201C] = "open",
  [0x3008] = "open",     [0x300A] = "open",
  [0x300C] = "open",     [0x300E] = "open",
  [0x3010] = "open",     [0x3014] = "open",
  [0x3016] = "open",     [0x3018] = "open",
  [0x301A] = "open",     [0xFF08] = "open",
  [0xFF3B] = "open",     [0xFF5B] = "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",    [0xFF01] = "close",
  [0xFF09] = "close",    [0xFF0C] = "close",
  [0xFF0E] = "close",    [0xFF1A] = "close",
  [0xFF1B] = "close",    [0xFF1F] = "close",
  [0xFF3D] = "close",    [0xFF5D] = "close",
  [0x00B7] = "middle",   [0x2027] = "middle",
  [0x30FB] = "middle",
  [0x2013] = "dash",     [0x2014] = "dash",
  [0x2015] = "dash",     [0xFF0D] = "dash",
  [0x2026] = "ellipsis", [0x22EF] = "ellipsis",
}

```

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

`punct_class` Look up the group of a character.

```
local function punct_class (c, s)
  local override = M.punct_override[c]
  if override ~= nil then
    return override or 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
```

```
1715
```

(*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.

```
1716 function M.set_charset (kind, mode, ranges)
1717   local target = M.charsets[kind]
1718   if not target then
1719     return
1720   end
1721   if mode == "set" then
1722     target           = {}
1723     M.charsets[kind] = target
1724   end
1725   for _, range in ipairs(ranges) do
1726     local first = range[1]
1727     local last  = range[2] or first
1728     for code = first, last do
1729       if mode == "remove" then
1730         target[code] = nil
1731       else
1732         target[code] = true
1733       end
1734     end
1735   end
1736 end
1737
```

(*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.

```
1738 M.punct_class = function (c, s)
1739   return punct_class(c, s or M.settings[1])
1740 end
1741
1742 M.punct_styles = {
1743   quanjiao = { compress_adjacent = true,  ratio = 0.5,
1744                keep_open_close = true, hang = true },
1745   plain    = { compress_adjacent = false, ratio = 0.0,
1746                keep_open_close = true },
1747   banjiao  = { compress_adjacent = false, ratio = 0.0, keep_open_close = true,
1748                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.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 `copy_glyph` applies it, so the offset belongs to the font a glyph is taken from 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`.)

`copy_glyph` Copy a glyph node, changing only its font.

```
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 giving way to a sub-block font or to a fallback font. 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))
       and M.attr_font then
      local target = has_attribute(cur, M.attr_font)
      if target and target > 0 then
        local block = block_font(target, cur.char, settings_of(cur))
        if block then
```

```
1951            target = block
1952          end
1953        end
1954        if target and M.attr_punct_font and punct_class(cur.char) then
1955          local punct = has_attribute(cur, M.attr_punct_font)
1956          if punct and punct > 0 then
1957            target = punct
1958          end
1959        end
1960        if target and target > 0 and not has_char(target, cur.char) then
1961          target = fallback_font(target, cur.char, settings_of(cur)) or target
1962        end
1963        if target and target > 0 and target ~= cur.font then
1964          if nxt and nxt.id == GLYPH and nxt.font == cur.font
1965             and (nxt.char == 0
1966               or (nxt.char >= 0xF0000 and nxt.char <= 0x10FFFD)) then
1967            local after = nxt.next
1968            free_node(nxt)
1969            nxt = after
1970          end
1971          local g = copy_glyph(cur, target)
1972          g.next  = nxt
1973          if prev then
1974            prev.next = g
1975          else
1976            head = g
1977          end
1978          free_node(cur)
1979          cur = g
1980        end
1981      end
1982      prev = cur
1983      cur  = nxt
1984    end
1985    return head
1986  end
1987
```

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

`class_of_char` Map a character to `CJK` or `LATIN`.

```
1988  local function class_of_char (c, s)
1989    if is_cjk(c, s) then
1990      return "CJK"
1991    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 carries material of its own. A box of no size is passed over, so the boundary rules read the text on both sides of it as one run.

```
local function box_visible (n)
  return n.width ~= 0 or n.height ~= 0 or n.depth ~= 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))
    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))
    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, 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

```

last_punct

The same for the last visible character of a list.

```
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))
      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))
    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")
  inherit_attributes(k, source)
  k.kern = width
  if M.attr_inserted then
    set_attribute(k, M.attr_inserted, 1)
  end
  return k
end

```

(*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 of no size is passed over, so an empty box inside CJK text does not break the

handle_boundary

run. Decide what to do at the boundary between two visible nodes. Only source-generated glue is a candidate for replacement; other glue and penalties are left alone, a penalty being an explicit barrier and a discretionary opaque, so neither is bridged. A boundary that touches a suspended run is bridged by nothing, except that a mark keeps its break opportunity: `Verb` suspends the spacing, and the glue at a mark has no width, so restoring it there moves no ink. An opening mark is excluded, because a break after it is prohibited and the kinsoku pass cannot see the prohibition across a suspended node. Insert the CJK–CJK or CJK–Latin glue, or the compression kern for two adjacent punctuation marks. 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.

```
2151 M.normal_spaced = {}
2152
2153 function M.set_normal_spaced_chars (chars)
2154   local set = {}
2155   for _, code in utf8.codes(chars) do
2156     set[code] = true
2157   end
2158   M.normal_spaced = set
2159 end
2160
2161 local function normal_spaced (n)
2162   return n and n.id == GLYPH and M.normal_spaced[n.char] == true
2163 end
2164
2165 local function long_punct (n)
2166   return n and n.id == GLYPH and M.charsets.long[n.char] == true
2167 end
2168
2169 local PUNCT_BOUNDARY_GLUE = { width = 0, stretch = 0, shrink = 0 }
2170
2171 local function insert_boundary_glue (head, prev, prev_visible, prev_end,
2172                                      cur, cur_start, settings)
2173   if not prev_end or not cur_start then
2174     return head
2175   end
2176   if long_punct(prev) and long_punct(cur) then
2177     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)
    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)
    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
  local cur_mark  = cur_punct and cur.id == GLYPH
  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) and not normal_spaced(cur) then
    if 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.

```
local function restore_source_spaces (head)
  local prev = nil
  local cur  = head
  while cur do
    if cur.id == GLUE and cur.subtype ~= SPACESKIP
       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.

```
local function handle_source_space (head, glue, prev_end, cur,
                                    cur_start, settings)
  if glue.subtype ~= SPACESKIP 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

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 disabled(cur) and not disabled(prev_visible)
       and not occupied then
      local mark = prev_visible.id == GLYPH
        and punct_class(prev_visible.char, settings)
      if mark and mark ~= "open" then
        local g = make_glue(PUNCT_BOUNDARY_GLUE, prev_visible)
        if g then
          g.next = cur
          if prev then
            prev.next = g
          else
            head = g
          end
        end
      end
    end
    return head
  end
  if prev and prev.id == GLUE then
    if not is_inserted(prev) then
      return handle_source_space(
        head, prev, prev_end, cur, start_class, settings)
    end
  elseif prev and prev.id == PENALTY then
  elseif prev and prev.id == DISC then
  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 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
        prev_visible = nil
        prev_end     = nil
      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. 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

`next_glyph` it by a kern or a ligature at most, and a glue, a penalty, another discretionary or a formula ends the word.

```
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_marked_disc

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.

```
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
```

forbids_break

The rules: closing and middle punctuation must not start a line, opening punctuation must not end one, a run of dashes or ellipses stays together, and two adjacent marks are not separated. `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" then
    return true
  end
  if prev_mark == "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.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 of no size 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
```

Glue and kerns are break opportunities; only the ones LuaCJK inserted and the source spaces are considered.

is_break_point

```
local function is_break_point (n)
  if n.id == GLUE then
    return is_inserted(n) or n.subtype == SPACESKIP
  elseif n.id == KERN then
    return is_inserted(n)
  end
  return false
end

local function apply_kinsoku (head)
  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 == DISC then
      forbid_marked_disc(cur, last)
    elseif is_break_point(cur) and last then
      local nxt                  = next_visible(cur.next)
      local settings             = settings_of(last)
      local prev_mark, prev_char = edge_mark(last, true)
      local next_mark, next_char = nil, nil
      if nxt then
        next_mark, next_char = edge_mark(nxt, false)
      end
      local cj = settings.cj_strict and next_char and M.cj[next_char]
      if nxt and (no_break_mark(last) or no_break_mark(nxt)
                  or cj
                  or (settings.kinsoku
                      and not disabled(last) and not disabled(nxt)
                      and forbids_break(prev_mark, prev_char,
```

```
                                          next_mark, next_char, settings)))
      then
        if not (prev and prev.id == PENALTY and prev.penalty >= 10000) then
          local p = new_node("penalty")
          inherit_attributes(p, last)
          p.penalty = 10000
          if M.attr_inserted then
            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.

```
local function ink_shift (class, delta)
  if class == "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 > 0 then
    target = settings.punct_width
  end
  if target then
    return target
  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 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)
       and not disabled(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), delta)
          local p = new_node("penalty")
          inherit_attributes(p, cur)
          p.penalty = 10000
          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` The node types of the two kinds of math character noad.

`MATH_TEXT_CHAR`

```
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
```

rewrite_noad

Point the nucleus of one noad at the math family of its CJK family, 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.

```
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, family)
    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.

```
2728 local function bound_width (settings)
2729   local value = settings and settings.bound_width or 0
2730   if value > 0 then
2731     return value
2732   end
2733   return nil
2734 end
2735
```

(*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.

```
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 (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 + cs:byte(i + 1) + 108, i + 2
    end
    if b >= 251 and b <= 254 then
      return -(b - 251) * 256 - cs:byte(i + 1) - 108, i + 2
    end
    if b == 28 then
      local v = cs:byte(i + 1) * 256 + cs:byte(i + 2)
      if v >= 32768 then v = v - 65536 end
      return v, i + 3
    end
    if b == 255 then
      return (cs:byte(i + 1) * 256 + cs:byte(i + 2)) / 65536, i + 3
    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(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 then
```

```
      local k, dx = 1, 0
      if #stack % 2 == 1 then dx, k = stack[1], 2 end
      while k <= #stack - 3 do
        point(dx, stack[k])
        point(stack[k + 1], stack[k + 2])
        point(stack[k + 3], 0)
        dx = 0
        k = k + 4
      end
      stack = {}
    elseif b == 27 then
      local k, dy = 1, 0
      if #stack % 2 == 1 then dy, k = stack[1], 2 end
      while k <= #stack - 3 do
        point(stack[k], dy)
        point(stack[k + 1], stack[k + 2])
        point(0, stack[k + 3])
        dy = 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
        local last = (#stack - k == 3)
        if horiz then
          point(stack[k], last and stack[k + 3] or 0)
          point(stack[k + 1], stack[k + 2])
          point(0, 0)
        else
          point(last and stack[k + 3] or 0, stack[k])
          point(stack[k + 1], stack[k + 2])
          point(0, 0)
        end
        horiz = not horiz
        k = k + 3
      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`.)

`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.

```
local function hang_bounds (head)
```

```
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 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 lead, _, lead_glyph       = nil, nil, nil
    if nxt then
      lead, _, lead_glyph = edge_mark(nxt, false)
    end
    local w_end   = 0
    local w_start = 0
    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 lead_glyph and not disabled(lead_glyph) and lead == "open"
       and style_of(settings_of(lead_glyph)).hang then
      w_start = hang_amount(lead_glyph, "open", settings_of(lead_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.

```
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)) 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 trail[j].id == PENALTY) 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)) then
      break
    end
    local target = trail[j + 1]
    if target and target.id == GLUE then
      local p = new_node("penalty")
      inherit_attributes(p, glyph)
      p.penalty  = penalty
      p.next     = target
      glyph.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: the font first, because the classes depend on it; then the spacing, the fixed widths, the prohibitions, and the widow penalties. 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)
  head = rewrite_fonts(head)
  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`.)

`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 and then reads the discretionaries it has just created, one word at a time.

```
function M.hyphenate (head, tail)
  lang.hyphenate(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: spacing, font correction, math, and hyphenation, which is the only one that runs inside the line breaker rather than around it.

```
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`, the one both lua-ul and this package use.

`LuaCJKfntef.attr`
`LuaCJKfntef.settings`

The attribute that carries the number of a settings entry, and the entry itself: the inset at both ends and whether punctuation is skipped. A separate entry per call keeps two decorations in one paragraph apart.

```
⟨*lua-fntef⟩
LuaCJKfntef = LuaCJKfntef or {}
```

```
local M       = LuaCJKfntef
local GLYPH   = node.id("glyph")
local GLUE    = node.id("glue")
local LEADERS = 100
local KERN    = node.id("kern")
local HLIST   = node.id("hlist")
local RULE    = node.id("rule")

M.attr     = M.attr     or nil
M.settings = M.settings or {}
M.next_id  = M.next_id  or 1

```

(*End of definition for* `LuaCJKfntef.attr` *and* `LuaCJKfntef.settings`.)

`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.

```
local em_metrics = {}

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.

```
function M.new_settings (subtract, skip)
  local id       = M.next_id
  M.next_id      = id + 1
  M.settings[id] = { subtract = subtract, skip = skip }
  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`.)

`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 = g.width or 0
      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`.)

`replace_pair` Replace the leaders glue of one decoration. The pieces are kerns and shorter leaders whose total width stays ⟨*w*⟩, so the text does not move; the negative glue that follows the leaders is kept and becomes the tail of the replacement.

```
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)
  local rule, down = box_rule(leaders.leader)
  local first, last
  local cursor = 0
  local function append (node_)
    if last then
      last.next = node_
    else
      first = node_
    end
    last = node_
  end
  for _, part in ipairs(parts) do
    if part[1] > cursor then
      local k = node.new("kern")
      k.kern  = part[1] - cursor
      append(k)
    end
    if rule then
      local r  = node.copy(rule)
      r.width  = part[2] - part[1]
```

```
      r.height = rule.height - down
      r.depth  = rule.depth + down
      r.next   = nil
      append(r)
    else
      local copy   = node.copy(leaders)
      copy.width   = part[2] - part[1]
      copy.stretch = 0
      copy.shrink  = 0
      copy.next    = nil
      append(copy)
    end
    cursor = part[2]
  end
  if cursor < w then
    local k = node.new("kern")
    k.kern  = w - cursor
    append(k)
  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.

```
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 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 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`.)

`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.

```
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)
    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, and the keys that lua-ul receives.

```
\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
\bool_new:N \l__LuaCJK_fntef_silent_bool
\bool_new:N \l__LuaCJK_fntef_hidden_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_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
\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 35.*)

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 and hidden when `hidden` is on. Every command goes through this, so `textformat` and `hidden` mean the same thing everywhere.

```
\cs_new_protected:Npn \__LuaCJK_fntef_body:n #1
  { \l__LuaCJK_fntef_textformat_tl \__LuaCJK_fntef_hide:n {#1} }
```

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

__LuaCJK_fntef_hide:n

The text of a decoration when `hidden` is on.

```
\cs_new_protected:Npn \__LuaCJK_fntef_hide:n #1
  {
    \bool_if:NTF \l__LuaCJK_fntef_hidden_bool
      { \phantom {#1} } { #1 }
  }
```

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

__LuaCJK_fntef_push_settings:

Hand the decoration settings to Lua, or clear the attribute when they are all at their defaults.

```
\cs_new_protected:Npn \__LuaCJK_fntef_push_settings:
  {
```

```
    \bool_lazy_and:nnTF
      { \dim_compare_p:nNn { \l__LuaCJK_fntef_subtract_dim } = { 0pt } }
      { \str_if_eq_p:Vn \l__LuaCJK_fntef_skip_tl { 0 } }
      {
        \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 } }
        \lua_now:e
          {
            LuaCJKfntef.new_settings
              ( \dim_to_decimal_in_sp:n { \l__LuaCJK_fntef_subtract_dim } ,
                \l__LuaCJK_fntef_skip_js_tl )
          }
      }
  }
\cs_new_protected:Npn \__LuaCJK_fntef_prepare_underline:n #1
  {
    \__LuaCJK_fntef_options:n {#1}
    \dim_set:Nn \l__LuaCJK_fntef_top_dim    { -\l__LuaCJK_fntef_depth_dim }
    \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 }
  }
```

__LuaCJK_fntef_formatted_underline:

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.

```
\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 38.*)

`__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,
    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
    \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`.)

`\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* 37.)

`\CJKunderwave` A wave drawn with l3draw, one period per half em, repeated by `\leaders`. 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 period is forced to exactly half an em by `\hbox to .5em`. 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 the ink fills the period exactly and 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. 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
```

```
\newunderlinetype \__LuaCJK_fntef_wave
  {
    \tex_leaders:D \hbox_to_wd:nn { .5em }
      {
        \tex_hss:D
        \hbox_set:Nn \l__LuaCJK_fntef_pattern_box
          {
            \box_move_up:nn { -.42em }
              {
                \hbox:n
                  {
                    \draw_begin:
                      \draw_set_linewidth:n { .04em }
                      \draw_path_moveto:n { 0em , 0em }
                      \draw_path_curveto:nnn { .0625em , .16em }
                        { .1875em , .16em } { .25em , 0em }
                      \draw_path_curveto:nnn { .3125em , -.16em }
                        { .4375em , -.16em } { .5em , 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
      }
  }
\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 37.*)

`\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.

```
\newunderlinetype \__LuaCJK_fntef_slash
  {
    \tex_leaders:D \hbox:n
      {
        \hbox_set:Nn \l__LuaCJK_fntef_pattern_box
          {
            \box_move_up:nn { -.17em }
              {
                \hbox:n
                  {
```

```
                    \draw_begin:
                      \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
      }
  }
\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 37.*)

`\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 37.*)

`\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* 37.)

`\__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` when it is given, 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 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.

```
\cs_new_protected:Npn \__LuaCJK_fntef_use_leader_type:nn #1#2
  {
```

```
    \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:Nn \l__LuaCJK_fntef_color_key_tl
          { format | \l__LuaCJK_fntef_format_tl }
      }
    \tl_set:Ne \l__LuaCJK_fntef_cache_key_tl
      { \exp_not:n {#1} | \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 }
        \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`.)

`\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 38.*)

`\CJKunderdot` Emphasise with dots below the glyphs. The optional argument takes the key set of `\CJKunderline`, like the other decorations.

```
\NewDocumentCommand \CJKunderdot { s t- o +m }
  {
    \group_begin:
      \__LuaCJK_fntef_decoration_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 dot pattern inside a group. One repeating unit is a dot of `.06em` with `.12em` on each side, lowered by `.4em`. The unit is written with `em` units so that it scales with the font size. The underline type that repeats one dot of the pattern.

```
\newunderlinetype \__LuaCJK_fntef_dots
  {
    \__LuaCJK_fntef_dots
    \tex_leaders:D \hbox:n
      {
        \hbox_set:Nn \l__LuaCJK_fntef_pattern_box
          {
            \skip_horizontal:n { .12em }
            \box_move_up:nn { -.4em }
              {
                \hbox:n
                  {
                    \tex_vrule:D width .06em height .06em
                      depth \c_zero_dim
                  }
              }
            \skip_horizontal:n { .12em }
          }
        \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
```

```
      }
  }

\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 38.*)

`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 optional argument places the box on the baseline: `t` for top, `c` for centered (the default), and `b` for bottom. A width of zero or less keeps the surrounding line width.

`\l__LuaCJK_fntef_fill_box` The box being filled, its requested width, the half height used for centring, and the requested baseline position.

`\l__LuaCJK_fntef_fill_width_dim`
`\l__LuaCJK_fntef_fill_half_dim`
`\l__LuaCJK_fntef_fill_position_tl`
`\l__LuaCJK_fntef_fill_tag_tl`
`\l__LuaCJK_fntef_fill_struct_bool`

```
\box_new:N \l__LuaCJK_fntef_fill_box
\dim_new:N \l__LuaCJK_fntef_fill_width_dim
\dim_new:N \l__LuaCJK_fntef_fill_half_dim
\tl_new:N \l__LuaCJK_fntef_fill_position_tl
\tl_new:N \l__LuaCJK_fntef_fill_tag_tl
\bool_new:N \l__LuaCJK_fntef_fill_struct_bool
```

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

```
\cs_new_protected:Npn \__LuaCJK_fntef_fill_begin:nn #1#2
  {
    \group_begin:
    \bool_set_false:N \l__LuaCJK_fntef_fill_struct_bool
    \cs_if_exist:NT \tag_struct_begin:n
      {
        \mode_if_vertical:TF
          {
            \tag_struct_begin:n { tag = text-block }
            \tl_set:Nn \l__LuaCJK_fntef_fill_tag_tl { text-block }
            \bool_set_true:N \l__LuaCJK_fntef_fill_struct_bool
          }
          { \tl_clear:N \l__LuaCJK_fntef_fill_tag_tl }
        \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 } }
      \cs_if_exist:NT \tag_mc_begin:n
        {
          \tl_if_empty:NTF \l__LuaCJK_fntef_fill_tag_tl
            { \tag_mc_begin:n { } }
            { \tag_mc_begin:n { tag = \l__LuaCJK_fntef_fill_tag_tl } }
        }
      \tex_noindent:D \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 }
        {
          \dim_set:Nn \l__LuaCJK_fntef_fill_half_dim
            { \box_ht_plus_dp:N \l__LuaCJK_fntef_fill_box }
          \box_set_ht:Nn \l__LuaCJK_fntef_fill_box { 0pt }
          \box_set_dp:Nn \l__LuaCJK_fntef_fill_box
            { \l__LuaCJK_fntef_fill_half_dim }
        }
        { c }
        {
          \dim_set:Nn \l__LuaCJK_fntef_fill_half_dim
            { \box_ht_plus_dp:N \l__LuaCJK_fntef_fill_box / 2 }
          \box_set_ht:Nn \l__LuaCJK_fntef_fill_box
            { \l__LuaCJK_fntef_fill_half_dim }
          \box_set_dp:Nn \l__LuaCJK_fntef_fill_box
            { \l__LuaCJK_fntef_fill_half_dim }
        }
      }
    \box_use_drop:N \l__LuaCJK_fntef_fill_box
    \cs_if_exist:NT \tag_struct_end:
      {
        \cs_if_exist:NT \tagpdfparaOn { \tagpdfparaOn }
        \bool_if:NT \l__LuaCJK_fntef_fill_struct_bool
```

```
          { \tag_struct_end: }
      }
    \group_end:
  }
\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.

```
⟨*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@setcjkflag{%
  \lst@ifletter
    \lst@Output
  \else
    \ifLuaCJK@listings@cjkrun\else\lst@OutputOther\fi
    \lst@lettertrue
  \fi
  \LuaCJK@listings@cjkruntrue}
\def\LuaCJK@listings@define#1#2{%
```

```
  \@tempcnta=#1\relax
  \loop\ifnum\@tempcnta<#2\relax
    \advance\@tempcnta\@ne
    \lccode`\~=\@tempcnta
    \lccode`\/=\@tempcnta
    \lowercase{\gdef~{\LuaCJK@listings@process/}}%
  \repeat}
\def\LuaCJK@listings@catcodes#1#2{%
  \@tempcnta=#1\relax
  \loop\ifnum\@tempcnta<#2\relax
    \advance\@tempcnta\@ne
    \catcode\@tempcnta=13\relax
  \repeat}
\def\LuaCJK@listings@ranges#1{%
  #1{"2E80}{"9FFF}%
  #1{"F900}{"FAFF}%
  #1{"FF00}{"FFEF}%
  #1{"AC00}{"D7AF}}
\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}
```

`\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 }
      } { Opt }
  }
⟨/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

\/ 4303
\\ 15, *77*, 135, 684, 2694
\~ 4302

A

\addCJKfontfeatures *9*, *27*, *63*, *185*, 1159
adjacent commands:
adjacent_marks 2419
ADJUST 1404
adjust 3566
\advance 4288, 4301, 4309
\allocationnumber 29, 3614
any commands:
any_cjk_math_enabled 2601
apply commands:
apply_check_single 3256
\apply_fixed_widths *131*, *132*
apply_fixed_widths 2535
apply_kinsoku 2394

B

\baselineskip *10*, *28*, *73*, 1073, 1085, 1090
\bfdefault *10*, *28*, *70*, 1079, 1081
\bfseries *71*
block commands:
block_font 1862
\boldmath *10*, *28*, *71*, *188*
bool commands:
\bool_gset_false:N 609
\bool_gset_true:N 607, 1374
\bool_if:NTF . . . 55, 60, 122, 124, 126, 142, 223, 376, 378, 635, 650, 652, 742, 915, 1010, 1098, 1141, 1382, 3665, 3973, 3981, 4039, 4267
\bool_lazy_and:nnTF . . . 1075, 3670
\bool_lazy_or:nnTF 845, 855
\bool_new:N 38, 39, 89, 90, 91, 130, 202, 356, 358, 426, 443, 444, 677, 678, 679, 804, 805, 806, 807, 886, 999, 3625, 3626, 4208
\bool_set_eq:NN . 731, 733, 748, 750
\bool_set_false:N 114, 115, 116, 217, 564, 603, 754, 837, 936, 1048, 1102, 3916, 4212
\bool_set_true:N 46, 51, 98, 103, 108, 144, 221, 367, 369, 561, 599, 737, 827, 828, 829, 830, 934, 1012, 1100, 3915, 4219, 4382, 4388, 4394, 4426
bound commands:
bound_width 2728
box commands:
\box_ht:N 4016
\box_ht_plus_dp:N 4248, 4256

\box_move_up:nn 3949, 4026, 4060, 4106, 4181
\box_new:N 680, 3938, 4203
\box_set_dp:Nn 3966, 4021, 4074, 4192, 4250, 4259
\box_set_ht:Nn 3965, 4019, 4073, 4191, 4249, 4257
\box_use_drop:N 3967, 4028, 4075, 4193, 4263
box_visible 2000
\box_wd:N 703, 4007

C

\catcode 4310
\ccwd 4364, 4366, 4368, 4371
\cdp 4364, 4373
cff commands:
cff_box 2963
cff_dict 2892
cff_index 2873
cff_load 2939
cff_string 2888
char commands:
\char_set_catcode_ignore:n 1322, 1324, 1326
\cht 4364, 4372
\CJKfamily .. *4*, *6*, *21*, *23*, *185*, 1260, 1288, 1289, 1290, 1291, 1292, 1293
\CJKfamilydefault *7*, *24*, *68*, *74*, *188*, 982, 1147, 1380, 1381
CJKfilltwosides (env.) 1, 1, 4203
CJKfilltwosides* (env.) 4275
\CJKfontspec *6*, *23*, *185*, *188*, 1277
\CJKrmdefault *7*, *24*, *25*, *68*, *74*, *188*, 1147, 1197, 1381
\CJKsfdefault *7*, *24*, 1147, 1202
\CJKsout *19*, *38*, 3751
\CJKttdefault *7*, *24*, 1147, 1207
\CJKunderanyline *18*, *37*, *185*, 4095
\CJKunderanysymbol .. *19*, *38*, *185*, 4153
\CJKunderdblline *18*, *37*, *185*, 3930
\CJKunderdot *19*, *38*, *185*, 4162
\CJKunderline *16*, *17*, *35*, *160*, 3618, 3739
\CJKunderwave *18*, *37*, *185*, 3938
\CJKxout *18*, *37*, *185*, 4054
class commands:
class_of_char 1988
class_of_node 1995
clist commands:
\clist_const:Nn 1317
\clist_gput_right:Nn 170
\clist_if_in:NnTF 169, 184
\clist_map_inline:Nn 218, 1114, 1321
\clist_map_inline:nn 22, 180, 290, 784, 835, 906, 1147, 1294, 3921, 4353, 4364, 4430
\clist_new:N 164, 177
\clist_put_right:Nn 185, 1254
\clist_remove_all:Nn 187
copy commands:
copy_glyph 1922
cs commands:
\cs_generate_variant:Nn 453, 454, 813, 1232, 4439
\cs_gset_protected:Npx 768
\cs_if_exist:NTF 18, 766, 1393, 4213, 4222, 4230, 4241, 4264, 4266, 4348
\cs_if_exist_p:N 1078
\cs_new:Npn 12, 631, 4086, 4478
\cs_new_eq:NN 4360, 4362, 4363
\cs_new_protected:Npn 43, 48, 53, 80, 178, 204, 210, 229, 231, 237, 249, 266, 274, 287, 312, 318, 336, 342, 381, 633, 687, 757, 774, 779, 814, 821, 823, 833, 864, 887, 902, 931, 938, 962, 980, 986, 1008, 1053, 1064, 1096, 1105, 1110, 1164, 1222, 1233, 1238, 1288, 1289, 1290, 1291, 1292, 1293, 1304, 1319, 1328, 1330, 1337, 1345, 3644, 3646, 3652, 3661, 3663, 3668, 3692, 3709, 3737, 3751, 3818, 3911, 3925, 3971, 3979, 3984, 4118, 4196, 4209, 4238, 4358, 4361, 4365, 4367, 4369, 4379, 4385, 4391, 4418, 4434
\cs_set_protected:Npn 4405
\cs_set_protected:Npx 1266
\cs_to_str:N 1265, 1371, 1372

\cs_undefine:N 4355
ctex commands:
\ctex_ *165*
\ctex_add_to_selectfont:n 4358, 4431
\ctex_at_end:n 4403
\ctex_define:n ... 4441, 4446, 4455
\ctex_if_ccglue_touched: ... 4397
\ctex_if_ccglue_touched:TF . 4407
\ctex_ltj_set_kanjiskip:N .. 4379
\ctex_ltj_set_xkanjiskip:N 4385, 4463
\ctex_ltj_zero_globaldefs: . 4360
\ctex_set_jfm:n 4434, 4439, 4451, 4467
\ctex_update_ccglue: 4391
\ctex_update_ccwd: 4365, 4408
\ctex_update_em_unit: 4367
\ctex_update_kanjisize: 4369, 4430
\ctex_update_xkanjiskip: 4418, 4430
ctex internal commands:
\l__ctex_ccglue_skip 4377, 4378, 4393, 4400
\l__ctex_jfm_tl 4436, 4440
\l__ctex_punct_tl . 4450, 4451, 4467
__ctex_update_stretch_auxii: 4405
__ctex_update_stretch_- auxiii: 4408
\l__ctex_xkanjiskip_skip 4414, 4415, 4421, 4423, 4425, 4462, 4463
\l__ctex_xkanjiskip_tl 4416, 4417, 4423, 4461
\cwd 4364, 4371

D

\def .. 4285, 4290, 4298, 4306, 4312, 4317
\defaultCJKfontfeatures *9*, *27*, *185*, 1157
dim commands:
\dim_compare:nNnTF . 691, 708, 4228
\dim_compare_p:nNn 3671
\dim_eval:n 717, 956, 1084, 1089, 3774, 3779, 3793, 3798, 3829, 3830, 3845, 3851, 4004, 4012
\dim_gset:Nn 702
\dim_new:N ... 427, 429, 672, 673, 674, 3618, 3619, 3622, 3627, 3628, 3635, 3636, 3637, 3638, 3939, 3940, 3941, 4096, 4204, 4205, 4364
\dim_set:Nn 428, 430, 532, 547, 689, 706, 3640, 3641, 3642, 3643, 3695, 3696, 3704, 3706, 3866, 3870, 3917, 3918, 3919, 3920, 3923, 3986, 3987, 4002, 4010, 4101, 4225, 4229, 4247, 4255, 4366, 4368, 4371, 4372, 4373
\dim_set_eq:NN 695
\dim_to_decimal_in_sp:n . 151, 160, 646, 647, 955, 3687, 3725, 3789
\dim_use:N 716, 3700, 3701, 3714, 3715, 3756, 3757, 3823, 3824, 3838, 3839, 3840, 4014, 4037, 4038, 4107
\dim_zero:N ... 531, 546, 3869, 3913
\c_zero_dim 3965, 3966, 4020, 4022, 4073, 4074, 4186, 4191, 4192, 4373
disabled 1830
DISC 1404
draw commands:
\draw_begin: 3953, 4064
\draw_end: 3961, 4069
\draw_path_curveto:nnn 3956, 3958
\draw_path_lineto:n 4067
\draw_path_moveto:n ... 3955, 4066
\draw_path_use_clear:n 3960, 4068
\draw_set_linewidth:n . 3954, 4065

E

edge commands:
edge_mark 2443
\else 4293, 4294, 4318
environments:
CJKfilltwosides 1, 1, 4203
CJKfilltwosides* 4275
exp commands:
\exp_args:Nc 28, 837, 1107, 1149, 1150, 3724, 3770, 3787, 4140, 4144
\exp_args:NV . 272, 699, 898, 984, 994, 1016, 1039, 1122, 1161, 1196,

1201, 1206, 1268, 1283, 1285, 1334, 3745, 3833, 3858, 4042, 4113
\exp_args:Nv 1297
\exp_not:N 771, 1267, 3828, 3844, 3998, 4000, 4002, 4004, 4007, 4010, 4012, 4015, 4016, 4019, 4020, 4021, 4022, 4023, 4024, 4025, 4026, 4027, 4028, 4029, 4030, 4031, 4104, 4106, 4108
\exp_not:n 282, 285, 770, 850, 860, 1170, 1171, 1267, 3906, 4001, 4108, 4131
\expandafter 1106
\ExplFileVersion 11

F

fallback commands:
fallback_font 1907
\fangsong 6, 24, 188, 1288, 4354
\fi 4294, 4296, 4321, 4326, 4330
file commands:
\file_if_exist:nTF 1339
\file_input:n 1340
find commands:
find_cjk_math 2661
first commands:
first_punct 2039
first_visible_class 2004
\font 1082, 1087, 1092
font commands:
font_advance 2855
font_box 3114
font_cmap 2786
font_file 2747
font_files 2736
font_glyph 2823
\fontfamily 699, 908, 1016, 1039, 1070, 1122
\fontseries 1071, 1081
\fontshape 1072
\fontsize 1073, 1083, 1088
fontspec commands:
\fontspec_gset_family:Nnn 254, 813, 873, 1178, 1244
forbid commands:
forbid_marked_disc 2412
forbids commands:
forbids_break 2425
fp commands:
\fp_new:N 357
\fp_set:Nn 104, 368
\fp_use:N 125, 377
free commands:
free_node 1404

G

\gdef 4304
\global 1082, 1087, 1092, 4320
GLUE 1404
GLYPH 1404
group commands:
\group_begin: 253, 667, 696, 872, 904, 1015, 1038, 1069, 1121, 1177, 1243, 3741, 3812, 3932, 4047, 4080, 4099, 4155, 4164, 4198, 4211
\group_end: .. 258, 670, 705, 881, 929, 1022, 1043, 1093, 1126, 1185, 1257, 3749, 3816, 3936, 4052, 4084, 4116, 4160, 4172, 4201, 4270

H

handle commands:
handle_boundary 2151
handle_source_space 2273
hang commands:
hang_amount 3182
hang_bounds 3195
has commands:
has_attribute 1404
has_char 1893
has_char_cache 1893
hbox commands:
\hbox:n 3951, 3998, 4056, 4062, 4104, 4108, 4176, 4183
\hbox_set:Nn 701, 3947, 4000, 4058, 4178
\hbox_to_wd:nn 3944
\heiti 6, 24, 188, 1288, 4354
HLIST 1404

hook commands:
\hook_gput_code:nnn 20, 664, 790, 792, 1138, 1139, 1296, 1299, 1378, 1391, 4350, 4359

I

i16 2738
\IfBooleanT 140
\IfNoValueTF 1265
\ifnum 4300, 4308
\ifx 4324, 4328
inherit commands:
inherit_attributes 2101
INS 1404
insert commands:
insert_boundary_glue 2151
int commands:
\int_compare:nNnTF 725, 889, 1058, 1112
\int_const:Nn 29, 1006, 3614
\int_eval:n 910
\int_from_hex:n 272
\int_gincr:N 1057, 1280, 3721, 3765, 4135
\int_gset:Nn 1005, 1018, 1041, 1124
\int_gzero:N 1117
\int_new:N 433, 802, 1000, 1003, 1004, 1109, 1277, 3630
\int_set:Nn 434, 574, 1067
\int_step_inline:nnn .. 1323, 1325
\int_use:N 29, 649, 891, 922, 1021, 1131, 1133, 1282, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 3614, 3617, 3677, 3723, 3767, 3992, 4091, 4137
\c_zero_int 725, 995, 997, 1026, 1033, 1046
is commands:
is_break_point 2477
is_inserted 2096
ITALICCORR 1404

K

\kaishu 6, 24, 188, 1288, 4354
KERN 1404
keys commands:
\keys_define:nn 94, 359, 457, 3861, 4457, 4470
\keys_if_exist:nnTF 3648
\l_keys_key_str 620, 3904
\keys_set:nn 117, 370, 627, 3645, 3649, 3650, 4477
\l_keys_value_tl 3905, 3906
keyval commands:
\keyval_parse:NNn .. 235, 838, 3657
kind 3416

L

last commands:
last_punct 2057
last_visible_class 2021
\lccode 4302, 4303
LIGATURE 1404
\lishu 6, 24, 188, 1288, 4354
LOCAL commands:
LOCAL_PAR 1404
\loop 4300, 4308
\lowercase 4304
\ltjgetparameter 4478
\ltjsetparameter 4476
lua commands:
\lua_escape:n 121, 151, 159, 174, 375, 641, 654, 1132, 1309
\lua_load_module:n 1359, 3612
\lua_now:n 118, 148, 156, 171, 315, 321, 339, 345, 353, 372, 637, 919, 951, 1128, 1306, 1329, 1360, 3616, 3674, 3684, 3989, 4088
LuaCJK 1396
LuaCJK internal commands:
__LuaCJK_add_fallback:nn 1217, 1218, 1220, 1238, 1238
__LuaCJK_add_family_features:n 1161, 1164, 1164
\l__LuaCJK_auto_fallback_bool 416, 599, 603, 652
\l__LuaCJK_autofake_bold_-given_bool 804, 827, 846
\g__LuaCJK_autofake_bold_tl . 389, 400, 408, 416, 593, 594, 843, 850

\l__LuaCJK_autofake_slant_-given_bool 804, 828, 856
\g__LuaCJK_autofake_slant_tl 393, 401, 412, 416, 614, 615, 853, 860
\g__LuaCJK_block_fontid_int 1109, 1117, 1124, 1133
\l__LuaCJK_block_key_tl 196, 1116, 1119
\g__LuaCJK_block_names_clist 163, 218, 1114
\g__LuaCJK_block_prop 163
\l__LuaCJK_blocks_off_clist 177, 184, 185, 187, 654
\l__LuaCJK_bold_font_given_-bool 804, 829, 847
__LuaCJK_bool_js:N 631, 642, 643, 644, 645, 648
\l__LuaCJK_bound_width_-warned_bool 130, 142, 144
\l__LuaCJK_break_between_-puncts_bool . . 416, 561, 564, 650
__LuaCJK_build_features:n 816, 821, 833
__LuaCJK_build_ranges:n 167, 287, 314, 320, 338, 344
\l__LuaCJK_ccglue_skip 36, 45, 57, 728, 744, 1371, 4378, 4381, 4393, 4400, 4482
\l__LuaCJK_ccglue_touched_-bool 38, 46, 55, 732, 748, 4382, 4394
\l__LuaCJK_check_single_tl 416, 569, 571, 648
\l__LuaCJK_cj_strict_tl 416, 481, 484, 643
\l__LuaCJK_cjk_math_tl 416, 490, 492, 645, 988
\g__LuaCJK_cjk_space_attr 22, 462, 463
\c__LuaCJK_cjk_space_attr_int 22, 1363
\l__LuaCJK_default_feature_tl 804, 840, 1158
\g__LuaCJK_disabled_attr 22, 660, 662
\c__LuaCJK_disabled_attr_int 22, 1370
\l__LuaCJK_ecglue_skip 36, 50, 62, 730, 746, 1372, 4387, 4413, 4415, 4421, 4425, 4483
\l__LuaCJK_ecglue_touched_-bool 39, 51, 60, 734, 750, 4388, 4426
\g__LuaCJK_embolden_factor_tl 390, 407, 416, 592
__LuaCJK_extract_font_-keys:nn 196, 237, 1235
\l__LuaCJK_fallback_family_tl 416, 1245, 1249, 1254
\l__LuaCJK_fallback_ids_tl 416, 905, 911, 912, 917, 923
\l__LuaCJK_fallback_key_tl 446, 1213, 1252, 1256
\g__LuaCJK_fallback_prop 416, 892, 1251, 1255
\g__LuaCJK_fallback_pushed_-prop 885
\g__LuaCJK_family_block_prop 196, 1118
\l__LuaCJK_family_default_-init_tl 1147, 1380
\l__LuaCJK_family_name_tl 798, 1013, 1016, 1021, 1027
\g__LuaCJK_family_prop 697, 795, 875, 882, 964, 983, 993, 1013, 1035, 1186, 1384
\l__LuaCJK_family_tl 697, 795, 882, 892, 897, 926, 966, 991, 1013, 1116, 1162
\l__LuaCJK_feature_tl 812, 817, 840, 842, 849, 859, 868, 869, 874, 1166, 1168, 1170, 1179, 1189
\l__LuaCJK_features_tl 47, 196, 1236, 1242, 1245
\l__LuaCJK_fntef_any_code_tl 3826, 3834, 3842, 3859, 3996, 4043, 4095, 4102, 4114
\l__LuaCJK_fntef_any_depth_-dim 3986, 4015, 4038, 4095, 4101, 4107

__LuaCJK_fntef_body:n . 3661, 3661, 3746, 3748, 3815, 3935, 4051, 4083, 4115, 4159, 4167, 4170
\l__LuaCJK_fntef_bottom_dim 3618, 3696, 3701, 3707
\l__LuaCJK_fntef_cache_key_tl 3631, 3711, 3719, 3733, 3753, 3763, 3806, 3820, 3834, 3835, 3859, 4034, 4043, 4111, 4114, 4130, 4133, 4149
\l__LuaCJK_fntef_cache_name_tl 3632, 3719, 3722, 3724, 3733, 3735, 3763, 3766, 3771, 3788, 3806, 3808, 4133, 4136, 4141, 4145, 4149, 4151
\l__LuaCJK_fntef_color_key_tl 3618, 4122, 4127, 4131
__LuaCJK_fntef_decoration_setup:n 3813, 3861, 3925, 3933, 4048, 4081, 4100, 4156, 4165
\l__LuaCJK_fntef_depth_dim 3618, 3695, 3697, 3823, 3829, 3831, 3838, 3847, 3853, 3885, 3917
__LuaCJK_fntef_dots . . 4174, 4199
\l__LuaCJK_fntef_em_descent_dim 3941, 3987, 4014, 4037
__LuaCJK_fntef_fill_begin:nn 4209, 4273, 4276
\l__LuaCJK_fntef_fill_box 4203, 4227, 4248, 4249, 4250, 4256, 4257, 4259, 4263
__LuaCJK_fntef_fill_end: 4238, 4274, 4277
\l__LuaCJK_fntef_fill_half_dim 4203, 4247, 4251, 4255, 4258, 4260
\l__LuaCJK_fntef_fill_position_tl 4203, 4224, 4243
\l__LuaCJK_fntef_fill_struct_bool 4203, 4212, 4219, 4267
\l__LuaCJK_fntef_fill_tag_tl 4203, 4218, 4221, 4232, 4234
\l__LuaCJK_fntef_fill_width_dim 4203, 4225, 4228, 4229
\g__LuaCJK_fntef_format_int 3630, 3721, 3723, 3765, 3767, 4135, 4137
\g__LuaCJK_fntef_format_prop 3629, 3718, 3732, 3762, 3805, 4132, 4148
\l__LuaCJK_fntef_format_tl 3621, 3713, 3717, 3743, 3758, 3760, 3761, 3768, 3893, 3894, 4120, 4126, 4128, 4138
__LuaCJK_fntef_formatted_underline: 3709, 3709, 3748
\l__LuaCJK_fntef_gap_dim 3637, 3642, 3840, 3849, 3855, 3889, 3919
\l__LuaCJK_fntef_height_dim 3638, 3643, 3756, 3776, 3781, 3795, 3800, 3891, 3920
\l__LuaCJK_fntef_hidden_bool 3626, 3665, 3881, 3916
__LuaCJK_fntef_hide:n 3662, 3663, 3663
__LuaCJK_fntef_option_bare:n 3644, 3646, 3658
__LuaCJK_fntef_option_pair:nn 3644, 3644, 3658
__LuaCJK_fntef_options:n 3644, 3652, 3694, 3927
\l__LuaCJK_fntef_pass_tl 3618, 3698, 3746, 3902
\l__LuaCJK_fntef_pattern_box 3938, 3947, 3965, 3966, 3967, 4000, 4007, 4016, 4019, 4021, 4028, 4058, 4073, 4074, 4075, 4178, 4191, 4192, 4193
__LuaCJK_fntef_prepare_underline:n 3692, 3742
__LuaCJK_fntef_push_settings: 3668, 3668, 3703, 3928
__LuaCJK_fntef_reset_decoration: 3654, 3911
\l__LuaCJK_fntef_rule_depth_dim 3628, 3706, 3715, 3730
\l__LuaCJK_fntef_rule_height_dim 3627, 3704, 3714, 3729

\g__LuaCJK_fntef_settings_-attr 3613, 3615
\c__LuaCJK_fntef_settings_-attr_int 3614, 3617, 3677
__LuaCJK_fntef_silent_begin: 3971, 4025
\l__LuaCJK_fntef_silent_bool . 3618, 3883, 3915, 3973, 3981, 4039
__LuaCJK_fntef_silent_end: 3979, 4029
\l__LuaCJK_fntef_skip_js_tl 3624, 3682, 3683, 3688
\l__LuaCJK_fntef_skip_tl 3623, 3672, 3681, 3877, 3878, 3914
__LuaCJK_fntef_slash . 4054, 4082
\l__LuaCJK_fntef_subtract_dim 3622, 3671, 3687, 3866, 3869, 3870, 3913
\l__LuaCJK_fntef_symbol_kern_-dim 3939, 4002, 4024, 4031
\l__LuaCJK_fntef_symbol_-raise_dim 3940, 4010, 4027
\l__LuaCJK_fntef_symbol_sep_-dim 3897, 3923
\l__LuaCJK_fntef_symbol_tl 3639, 3896, 4001, 4040, 4049, 4157, 4166
\l__LuaCJK_fntef_textformat_-tl 3618, 3662, 3899
\l__LuaCJK_fntef_thickness_-dim 3618, 3697, 3757, 3777, 3782, 3796, 3801, 3824, 3831, 3839, 3848, 3854, 3887, 3918
\l__LuaCJK_fntef_top_dim 3618, 3695, 3700, 3705
__LuaCJK_fntef_underdot:n 4167, 4174, 4196
__LuaCJK_fntef_underLine:nn 3737, 3745
__LuaCJK_fntef_use_double: 3818, 3818, 3934
__LuaCJK_fntef_use_leader_-type:nn 3833, 3858, 4042, 4113, 4118, 4118
__LuaCJK_fntef_use_sout: 3751, 3814
__LuaCJK_fntef_use_symbol:n 3984, 4050, 4158, 4169
__LuaCJK_fntef_wave . . 3942, 4050
\g__LuaCJK_font_attr 22, 1023, 1026, 1388
\c__LuaCJK_font_attr_int 22, 1362, 3992, 4091
\g__LuaCJK_font_feature_prop 804, 817, 868, 1166, 1188
\g__LuaCJK_font_name_prop 804, 818, 866, 1173
__LuaCJK_font_option:n 229, 235
__LuaCJK_font_option:nn 210, 235
\g__LuaCJK_font_shift_prop 416, 878, 938, 1182, 1248
\g__LuaCJK_fontid_int 889, 891, 922, 999, 1018, 1021, 1023, 1112, 1131, 1388
\l__LuaCJK_fontid_tl 416, 891, 895, 926
\g__LuaCJK_fontspec_int 1277, 1280, 1282
\l__LuaCJK_fontspec_tl 798, 873, 875, 879, 1178, 1183, 1187
\l__LuaCJK_hex_int_tl 261, 271, 282
__LuaCJK_hex_store:n . . 266, 280
\l__LuaCJK_hex_tl 261, 268, 269, 270, 272
__LuaCJK_ignore_format_-chars: 1317, 1319, 1377
\c__LuaCJK_ignored_chars_-clist 1317, 1321
\g__LuaCJK_inserted_attr 22
\c__LuaCJK_inserted_attr_int 22, 1365
\l__LuaCJK_is_block_bool 202, 217, 221, 223
\l__LuaCJK_italic_font_given_-bool 804, 830, 857
\l__LuaCJK_latin_punct_tl 422, 423, 552, 554, 644
__LuaCJK_load_cj: 485, 1328, 1328

__LuaCJK_load_config_file:n 1334, 1337
__LuaCJK_load_fandol: 1345, 1345, 1385
\g__LuaCJK_load_fandol_bool 416, 607, 609, 1382
__LuaCJK_load_local_config: 1330, 1330, 1390
\l__LuaCJK_local_config_tl 416, 579, 580, 581, 1332, 1334
\g__LuaCJK_lua_loaded_bool 635, 885, 1374
__LuaCJK_math_family:n 1053, 1066
\g__LuaCJK_math_family_int 1004, 1057, 1058, 1061
\c__LuaCJK_math_family_max_int 1004, 1058
\g__LuaCJK_math_family_prop 1004, 1055, 1060, 1068
\l__LuaCJK_math_family_tl 990, 992, 1001, 1274
\g__LuaCJK_math_font_attr 22, 995, 997, 1094
\c__LuaCJK_math_font_attr_int 22, 1366
__LuaCJK_math_font_update: 986, 986, 1029, 1101
__LuaCJK_math_version_refresh: ... *68*, 1096, 1096, 1108
\g__LuaCJK_nobreak_attr 22, 668, 736, 753
\c__LuaCJK_nobreak_attr_int 22, 1368
\g__LuaCJK_normal_spaced_chars_tl 1301, 1309, 1314
__LuaCJK_note_feature_key:n 822, 823, 839
__LuaCJK_note_feature_key:nn 821, 839
\l__LuaCJK_option_key_tl 196
\l__LuaCJK_option_seq 196
__LuaCJK_parse_font_options:n 196, 231, 240, 1240
\g__LuaCJK_pending_block_prop 196
\l__LuaCJK_punct_bound_width_dim 416, 546, 547, 647
\l__LuaCJK_punct_compress_bool 356, 361, 367, 376
\l__LuaCJK_punct_compress_given_bool 89, 98, 114, 122
\l__LuaCJK_punct_compress_value_tl 92, 99, 123
\l__LuaCJK_punct_family_tl 538, 539, 1001, 1032, 1035
\g__LuaCJK_punct_font_attr 22, 1033, 1044, 1046
\c__LuaCJK_punct_font_attr_int 22, 1367
\g__LuaCJK_punct_fontid_int 1001, 1041, 1044
\l__LuaCJK_punct_keep_bool 356, 363, 369, 378
\l__LuaCJK_punct_keep_given_bool 89, 108, 116, 126
\l__LuaCJK_punct_keep_value_tl 93, 109, 127
\l__LuaCJK_punct_kinsoku_tl 416, 475, 477, 642
\l__LuaCJK_punct_ratio_fp 104, 125, 356, 362, 368, 377
\l__LuaCJK_punct_ratio_given_bool 89, 103, 115, 124
\g__LuaCJK_punct_style_prop 66, 82, 371
\l__LuaCJK_punct_style_tl 73, 83, 86, 641
\l__LuaCJK_punct_width_dim 416, 531, 532, 646
__LuaCJK_push_block_fonts: 1031, 1109, 1110
__LuaCJK_push_class:nn 300, 301, 307, 308, 309, 318
__LuaCJK_push_fallback: 885, 887, 935, 1030
__LuaCJK_push_fallback_aux:n 898, 902
__LuaCJK_push_normal_space:n 306, 312

__LuaCJK_push_punct:nn ....
..... 302, 303, 304, 305, 328, 329, 330, 331, 332, 333, 336
__LuaCJK_push_shift:nn ....
..... 913, 938, 938, 1020
__LuaCJK_put_feature:n ....
..... 204, 226, 230
__LuaCJK_range_entry:n 274, 290
\l__LuaCJK_range_pair_tl ....
..... 261, 277, 281, 285
\l__LuaCJK_range_seq 261, 276, 278
\l__LuaCJK_ranges_tl 168, 174, 261, 284, 289, 316, 322, 340, 348
__LuaCJK_refresh_fallbacks:
..... 600, 885, 931, 1258
__LuaCJK_refresh_glues: 53, 1143
__LuaCJK_register_block_-
fonts:n 247, 249
__LuaCJK_register_family:n .
..... 819, 864, 864
__LuaCJK_register_math_-
family:n 994, 1051, 1064
\l__LuaCJK_registered_name_tl
803, 866, 867, 874, 1173, 1174, 1179
__LuaCJK_reset_family: ....
..... 980, 980, 1300
\l__LuaCJK_selectfont_bool ..
..... 934, 936, 999, 1010, 1012, 1048, 1098, 1100, 1102, 1141
__LuaCJK_set_autofake:nn ...
..... 381, 381, 584, 587
__LuaCJK_set_block_enabled:nn
..... 178, 193, 195
__LuaCJK_set_ccglue:n .....
..... 43, 43, 459, 4472
__LuaCJK_set_charset:nnn ...
... 342, 342, 505, 507, 509, 511, 513, 515, 517, 519, 521, 523, 525, 527
__LuaCJK_set_ecglue:n .....
..... 43, 48, 460, 4473
__LuaCJK_set_family:nnn ....
..... 795, 814, 1232, 1236
__LuaCJK_set_family_with_-
options:nnnn .....
..... 75, 1196, 1201, 1206, 1210, 1222, 1222, 1268, 1273, 1283
__LuaCJK_set_family_with_-
options_aux:nnn .....
..... 1227, 1228, 1230, 1233
__LuaCJK_set_normal_spaced_-
chars: 354, 1304, 1315, 1376
__LuaCJK_set_punct_style:n .
..... 73, 80, 471
\g__LuaCJK_settings_attr 22
\c__LuaCJK_settings_attr_int
..... 22, 1369
__LuaCJK_settings_push: ....
..... 189, 628, 631, 633, 752, 1144, 1375, 4383, 4389, 4395, 4427
\g__LuaCJK_shift_decl_prop ..
..... 241, 416, 870, 1175
\l__LuaCJK_shift_dim_tl ....
..... 416, 870, 871, 876, 879, 938, 1175, 1176, 1180, 1183
\l__LuaCJK_shift_family_tl ..
..... 416, 938
\l__LuaCJK_shift_fid_tl 416, 938
\g__LuaCJK_shift_pushed_prop
..... 416, 938
\l__LuaCJK_shift_tl .....
..... 47, 196, 1241, 1246, 1249
\g__LuaCJK_slant_factor_tl ..
..... 394, 411, 416, 613
__LuaCJK_switch_family:n . 63, 962, 962, 984, 1261, 1267, 1285, 1297
\l__LuaCJK_tmpa_int .....
.. 802, 1067, 1082, 1087, 1092, 1094
\l__LuaCJK_tmpa_tl 254, 257, 697, 699, 798, 892, 898, 982, 983, 984, 991, 992, 993, 1036, 1039, 1119, 1122, 1252, 1253, 1254, 1256, 1264, 1267, 1269, 1281, 1284, 1285
\l__LuaCJK_tmpb_tl 801, 895, 896, 897, 909, 911, 913, 993, 994
__LuaCJK_toks_put_right:Nn .
..... 1105, 1105, 1107
\g__LuaCJK_unknown_family_seq
..... 970, 972, 977

__LuaCJK_update_font: 65, 68, 73, 190, 490, 492, 540, 999, 1008, 1138, 1275, 1387
\l__LuaCJK_verb_addon_bool 672, 737, 742, 754
__LuaCJK_verb_addon_compute: 687, 735
\l__LuaCJK_verb_box . 672, 701, 703
\l__LuaCJK_verb_cjk_dim 672, 695, 702, 707
\l__LuaCJK_verb_gap_dim 672, 706, 708, 716, 717
\l__LuaCJK_verb_mode_tl . 416, 496, 498, 500, 502, 759, 762, 776, 781
\l__LuaCJK_verb_saved_ccglue_skip 672, 727, 745
\l__LuaCJK_verb_saved_ccglue_touched_bool 672, 731, 749
\l__LuaCJK_verb_saved_ecglue_skip 672, 729, 747
\l__LuaCJK_verb_saved_ecglue_touched_bool 672, 733, 751
\l__LuaCJK_verb_space_dim 672, 689, 695, 707
__LuaCJK_verbatim@font_hook: 757, 757, 771
__LuaCJK_verbatim_env_begin: 774, 774, 791
__LuaCJK_verbatim_env_end: 774, 779, 793
\c__LuaCJK_version_tl 11, 12
\l__LuaCJK_widow_penalty_int 416, 574, 649
\g__LuaCJK_xecglue_attr 22, 467, 468
\c__LuaCJK_xecglue_attr_int 22, 1364

LuaCJK.attr commands:

LuaCJK.attr_cjk_space 1427
LuaCJK.attr_disabled 1828
LuaCJK.attr_font 1427
LuaCJK.attr_inserted 1427
LuaCJK.attr_math_font 1427
LuaCJK.attr_settings 1435
LuaCJK.attr_xecglue 1427
LuaCJK.cj 1527

LuaCJK.cj commands:

LuaCJK.cj_count 1527
LuaCJK.cj_loaded 1527

LuaCJK.declare commands:

LuaCJK.declare_ranges 1667

LuaCJK.define commands:

LuaCJK.define_punct_style . . 1757
LuaCJK.hpack 3329
LuaCJK.hyphenate 3337

LuaCJK.is commands:

LuaCJK.is_cjk 1569

LuaCJK.new commands:

LuaCJK.new_settings 1455

LuaCJK.pre commands:

LuaCJK.pre_linebreak 3325
LuaCJK.pre_mlist_to_hlist . . 3333
LuaCJK.process 3311

LuaCJK.punct commands:

LuaCJK.punct_styles 1738

LuaCJK.set commands:

LuaCJK.set_charset 1716
LuaCJK.settings 1435

LuaCJK.settings commands:

LuaCJK.settings_by_signature 1435

LuaCJK.warned commands:

LuaCJK.warned_cjk_math 2659
\LuaCJKCancelSubCJKBlock 4, 7, 21, 25, 187, 177
\LuaCJKDeclareCharClass 5, 7, 12, 13, 22, 24, 31, 51, 186, 261
\LuaCJKDeclarePunctClass 5, 13, 22, 31, 186, 324
\LuaCJKDeclarePunctStyle 14, 20, 32, 39, 43, 92, 186, 356
\LuaCJKDeclareSubCJKBlock 5, 7, 22, 24, 187, 163
\LuaCJKdisable 15, 34, 187, 659, 760, 763, 777, 4329
\LuaCJKEditPunctStyle . 13, 32, 187, 89
\LuaCJKemdescent 18, 37, 187, 4086
\LuaCJKenable 15, 34, 187, 659, 782

LuaCJKfntef commands:

\LuaCJKfntef_use_symbol:n . . . 159
LuaCJKfntef.attr 3361

LuaCJKfntef.em commands:
LuaCJKfntef.em_descent 3376
LuaCJKfntef.hpack 3590
LuaCJKfntef.new commands:
LuaCJKfntef.new_settings ... 3408
LuaCJKfntef.settings 3361
\LuaCJKnobreak .....
... 15, 20, 33, 40, 110, 187, 188, 665
\LuaCJKOffVerbAddon . 16, 34, 187, 672
\LuaCJKResetCharClass 13, 32, 186, 351
\LuaCJKRestoreSubCJKBlock .....
..... 4, 7, 21, 25, 187, 177
\LuaCJKsetkern .. 5, 14, 22, 32, 187, 154
\LuaCJKsetup 4, 21, 53,
58, 62, 416, 711, 714, 721, 4226, 4437
\LuaCJKsetwidth .....
..... 5, 11, 14, 22, 30, 32, 187, 130
\LuaCJKShipoutHook .. 15, 34, 187, 663
\LuaCJKVerbAddon 16, 34, 187, 672
\LuaCJKversion 4, 21, 187, 11
luaul commands:
\luaul_apply_color: 3727, 3791, 4146
\luaul_context_color: . 3725, 3789
\luaul_set_color:n 3717, 3761, 4126

M

M 1404
M.block commands:
M.block_fonts 1838
M.block_order 1838
M.blocks 1838
M.charsets 1691
M.fallback 1881
M.yoffset 1887
make commands:
make_glue 2107
make_kern 2141
\makeatletter 4281
\makeatother 4331
MARK 1404
mark commands:
mark_blank 3151
MATH 1404
MATH commands:
MATH_CHAR 2598
MATH_TEXT_CHAR 2598
\mddefault 10, 28, 70, 1071
mode commands:
\mode_if_vertical:TF 4215
msg commands:
\msg_critical:nn .....
... 7, 19, 21, 3608, 4339, 4349, 4351
\msg_error:nn 1059
\msg_new:nnn .....
..... 5, 13, 75, 131, 292, 294,
623, 681, 978, 1051, 1343, 3606, 4337
\msg_new:nnnn 4342
\msg_warning:nn ... 145, 710, 4444
\msg_warning:nnn .....
.. 85, 310, 334, 619, 973, 1192, 1341

N

new commands:
new_node 1404
\newattribute 28, 3613
\newCJKfontfamily 6, 23, 185, 188, 1262
\NewDocumentCommand 112, 138,
154, 165, 192, 194, 296, 324, 351,
365, 625, 659, 661, 663, 665, 723,
740, 1157, 1159, 1194, 1199, 1204,
1209, 1211, 1260, 1262, 1271,
1278, 1312, 3739, 3810, 3930,
4045, 4078, 4097, 4153, 4162, 4476
\NewDocumentEnvironment .. 4272, 4275
\newfontfamily 6, 23
\newif 4282, 4283
\newunderlinetype 3724, 3770,
3787, 3942, 4054, 4140, 4144, 4174
next commands:
next_glyph 2402
next_visible 2461
no commands:
no_break_mark 2394
\normalfont 188
\normalspacedchars .....
.... 11, 13, 29, 31, 32, 51, 187, 1301

P

PENALTY 1404
\phantom 3666

\pingfang 4354
prg commands:
\prg_do_nothing: .. 4360, 4362, 4363
\prg_new_conditional:Npnn .. 4397
\prg_return_false: 4401
\prg_return_true: 4401
\ProcessKeyOptions 630
prop commands:
\prop_gclear:N 239, 933
\prop_get:NnNTF 453, 697, 866, 868, 870, 892, 894, 942, 993, 1013, 1035, 1118, 1166, 1173, 1175, 1251, 3718, 3762, 4132
\prop_gput:Nnn 67, 68, 69, 70, 71, 72, 168, 224, 241, 256, 371, 454, 817, 818, 875, 878, 925, 949, 1060, 1182, 1248, 1255, 3732, 3805, 4148
\prop_if_empty:NTF 242, 1384
\prop_if_in:NnTF 82, 882, 947, 964, 983, 1055
\prop_item:Nn 1068
\prop_map_inline:Nn 251
\prop_new:N 66, 163, 196, 197, 442, 447, 448, 449, 795, 808, 809, 885, 1007, 3629
\prop_put:Nnn 1186, 1188
PUNCT commands:
PUNCT_CLASS 1627
punct commands:
punct_class 1653
punct_style 1765

R

\relax 4299, 4300, 4307, 4308, 4310
\repeat 4305, 4311
replace commands:
replace_pair 3510
\RequirePackage 10, 1392, 1394, 3611, 4280, 4352
restore commands:
restore_source_spaces 2259
rewrite commands:
rewrite_fonts 1940
rewrite_math 2614
rewrite_math_list 2614
rewrite_noad 2616
\rule 18, 37

S

\scriptfont 1087
\scriptscriptfont 1092
segments 3424
\selectfont 700, 883, 908, 967, 1017, 1040, 1074, 1081, 1086, 1091, 1123, 1190
seq commands:
\seq_gput_right:Nn 972
\seq_if_in:NnTF 970
\seq_map_inline:Nn 278
\seq_new:N 201, 262, 977
\seq_set_split:Nnn 276
set commands:
set_attribute 1404
set_glue 2125
\setCJKfallbackfamilyfont 7, 25, 186, 1211
\setCJKfamilyfont . 4, 6, 12, 21, 23, 30, 185, 1194, 1352, 1354, 1356, 1357
\setCJKmainfont 4, 5, 21, 23, 74, 185, 1194, 1347
\setCJKmathfont 10, 27, 69, 117, 187, 1271
\setCJKmonofont 4, 5, 21, 23, 185, 1194, 1351
\setCJKsansfont 4, 5, 21, 23, 185, 1194, 1349
settings commands:
settings_at 3453
settings_of 1515
\shapedefault ... 10, 28, 70, 1072, 1079
single commands:
single_rule 3464
skip commands:
\skip_horizontal:n 4023, 4030, 4180, 4189
\skip_if_eq:nnTF 4399, 4420
\skip_new:N 36, 37, 675, 676, 4377, 4414
\skip_set:Nn 40, 41, 45, 50, 57, 62, 4413, 4423, 4462

\skip_set_eq:NN 727, 729, 744, 746, 4378, 4381, 4387, 4393, 4415, 4424
\skip_use:N 4482, 4483
\songti 6, 24, 188, 1288, 4354
SPACESKIP 1404
spacing 2151
str commands:
\c_backslash_str 1303
\str_case:nn 825, 4243
\str_case:nnTF 298, 326, 383, 577, 4480
\str_if_eq:nnTF 182, 214, 220, 387, 399, 405, 530, 537, 545, 632, 3681, 3865, 3868, 3876
\str_if_eq_p:nn 1076, 3672
SUB commands:
SUB_MLIST 2612
sys commands:
\sys_if_engine_luatex:TF 3, 3604, 4335

T

tag commands:
\tag_mc_begin:n . . 4230, 4233, 4234
\tag_mc_end: 4241
\tag_struct_begin:n . . . 4213, 4217
\tag_struct_end: 4264, 4268
\tagpdfparaOff 4222
\tagpdfparaOn 4266
TEX and LATEX 2ε commands:
\@empty 4324, 4328
\@ne 4288, 4301, 4309
\@tempcnta 4299, 4300, 4301, 4302, 4303, 4307, 4308, 4309, 4310
\addCJKfontfeatures 9, 27
\begin 11, 29
\char 11, 29, 79
\CJKfamily 6, 23
\CJKfontspec 6, 23
\CJKsout 19, 38
\CJKunderanyline 18, 37
\CJKunderanysymbol 19, 38
\CJKunderdblline 18, 37
\CJKunderdot 19, 38
\CJKunderline 16, 35
\CJKunderwave 18, 37
\CJKxout 18, 37
\color 147
\CTEX@alchar 4361
\CTEX@beginallalchar 4362
\CTEX@endallalchar 4363
\defaultCJKfontfeatures . . . 9, 27
\end 11, 29
\every@math@size 72
\f@size 4366, 4368, 4372
\glb@settings 72
\hbox 153
\hfill 107
\hpack 135
\hskip 106, 107
\hss 153
\ifLuaCJK@listings@cjkrun 4282, 4294
\ifLuaCJK@listings@defined 4283, 4318
\lccode 163
\leaders 18, 37, 137, 141, 147, 153, 157, 158
\lpcode 120
\lst@AddToHook . . . 4284, 4323, 4327
\lst@Append 4289
\lst@ifletter 4291
\lst@inlinechars 4324, 4328
\lst@length 4288
\lst@lettertrue 4295
\lst@Output 4292
\lst@OutputOther 4294
\lst@whitespacefalse 4286
\lstinline 16, 35, 164
\LuaCJK@listings@activate 4317, 4325
\LuaCJK@listings@catcodes 4306, 4322
\LuaCJK@listings@cjkrunfalse 4284
\LuaCJK@listings@cjkruntrue 4297
\LuaCJK@listings@define 4298, 4319
\LuaCJK@listings@definedtrue 4320
\LuaCJK@listings@process 4285, 4304

\LuaCJK@listings@ranges 4312, 4319, 4322
\LuaCJK@listings@setcjkflag 4287, 4290
\LuaCJKCancelSubCJKBlock . . . *7*, *25*
\LuaCJKDeclareCharClass . . *13*, *31*
\LuaCJKDeclarePunctClass . . *13*, *31*
\LuaCJKDeclarePunctStyle . . *14*, *32*
\LuaCJKDeclareSubCJKBlock . . *7*, *24*
\LuaCJKdisable *15*, *34*
\LuaCJKEditPunctStyle *13*, *32*
\LuaCJKenable *15*, *34*
\LuaCJKnobreak *15*, *33*
\LuaCJKOffVerbAddon *16*, *34*
\LuaCJKRestoreSubCJKBlock . . *7*, *25*
\LuaCJKsetkern *14*, *32*
\LuaCJKsetup *4*, *21*
\LuaCJKsetwidth *14*, *32*
\LuaCJKShipoutHook *15*, *34*
\math@version 1076
\mathversion *9*, *27*, *72*
\mbox *185*
\newCJKfontfamily *6*, *23*
\normalfont *7*, *24*, *68*, *74*
\normalspacedchars *11*, *29*
\p@ 4366, 4368, 4372
\protrudechars *120*
\raise *156*
\rightmarginkern *120*
\rmfamily *7*, *24*, *74*
\rpcode *120*
\selectfont *69*
\setCJKfallbackfamilyfont . . *7*, *25*
\setCJKfamilyfont *6*, *23*
\setCJKmainfont *5*, *23*
\setCJKmathfont *10*, *27*
\sffamily *7*, *24*
\the *72*
\ttfamily *7*, *24*
\underLine *20*, *39*
\verb *16*, *35*, *62*
\verbatim@font . . . *62*, 766, 768, 770
\XeTeXglyphbounds *120*
\xout *18*, *37*, *156*

tex commands:

\tex_attribute:D 3992, 4091
\tex_baselineskip:D 40, 58
\tex_currentgrouplevel:D 725
\tex_endinput:D 8, 3609, 4340
\tex_font:D 690, 692, 693, 910, 1019, 1042, 1125
\tex_fontcharwd:D 693
\tex_fontdimen:D 690, 692
\tex_fontid:D 910, 1019, 1042, 1125
\tex_hsize:D 4229
\tex_hss:D 3946, 3968
\tex_ignorespaces:D 1261, 1286, 4236
\tex_leaders:D 3728, 3773, 3792, 3828, 3844, 3944, 3998, 4056, 4104, 4176
\tex_noindent:D 4236
\tex_par:D 4240
\tex_pdfextension:D . . . 3975, 3982
\tex_the:D 1106, 4123
\tex_vrule:D 3728, 3773, 3792, 3828, 3844, 4185
\textbf *106*, *188*
\textcolor *17*, *36*
\textfont 1082
\textnormal *188*

tl commands:

\tl_clear:N . . 233, 234, 277, 289, 538, 580, 867, 869, 871, 896, 905, 1027, 1174, 1176, 1253, 3922, 4221
\tl_const:Nn 11
\tl_gclear:N 400, 401
\tl_gset:Nn 407, 408, 411, 412, 439, 441, 592, 594, 613, 615, 1302, 1314, 1381
\tl_gset_eq:NN 389, 393
\tl_if_blank:nTF 841, 1171
\tl_if_empty:NTF 206, 244, 593, 614, 843, 853, 876, 917, 945, 990, 1032, 1180, 1241, 1246, 1332, 3743, 3760, 3768, 3905, 4049, 4120, 4138, 4166, 4232
\tl_if_eq:NNTF 897, 1380
\tl_if_eq:NnTF 988
\tl_if_eq:nnTF . . 759, 762, 776, 781

\tl_if_novalue:nTF 1214, 1216, 1224, 1226, 3655
\tl_new:N 73, 92, 93, 198, 199, 200, 203, 261, 263, 264, 265, 416, 418, 420, 422, 424, 431, 435, 436, 437, 438, 440, 445, 446, 450, 451, 452, 455, 456, 796, 798, 799, 800, 801, 803, 810, 812, 1001, 1002, 1149, 1153, 1155, 1301, 3620, 3621, 3623, 3624, 3631, 3632, 3633, 3634, 3639, 4095, 4206, 4207, 4416, 4440
\tl_put_left:Nn 3698
\tl_put_right:Nn 72, 208, 281, 284, 842, 849, 859, 911, 912, 3902
\tl_replace_all:Nnn 269
\tl_set:Nn 74, 83, 86, 99, 109, 207, 212, 215, 245, 255, 268, 271, 417, 419, 421, 423, 425, 432, 475, 477, 481, 484, 490, 492, 496, 498, 500, 502, 539, 552, 554, 569, 571, 579, 581, 797, 811, 891, 909, 940, 941, 966, 982, 1116, 1150, 1154, 1158, 1168, 1213, 1242, 1264, 1274, 1281, 3682, 3683, 3711, 3722, 3753, 3766, 3820, 3826, 3835, 3842, 3877, 3878, 3914, 3996, 4034, 4102, 4111, 4122, 4127, 4130, 4136, 4157, 4218, 4224, 4417, 4436, 4450, 4461
\tl_set_eq:NN .. 840, 991, 992, 1156
\tl_to_str:n 4040, 4112
\tl_trim_spaces:N 213, 270
\tl_trim_spaces:n 825
\tl_use:N 12
token commands:
\token_to_str:N 133, 685
transparent commands:
transparent_node 1995
\typeout 4334

U

u16 2738
u32 2738
\underLine 3738
use commands:
\use:N 1073, 1079, 1084, 1089, 3735, 3808, 4123, 4151
\use_i:nn 1149, 1150
\use_ii:nn 1151
\usepackage 4, 21

V

vbox commands:
\vbox_set:Nw 4227
\vbox_set_end: 4242
\verb 186
VLIST 1404

W

warn commands:
warn_disabled_cjk_math 2661
WHATSIT 1404

Y

\yahei 4354
\youyuan 6, 24, 188, 1288, 4354

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. 134

apply_kinsoku: Provide PunctKinsoku line-breaking prohibitions: closing punctuation cannot start a line and opening punctuation cannot end a line. 110

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. 120

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. 161

\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. 157

\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. 153

copy_glyph: A character the font lacks leaves no placeholder: the .notdef or private-use glyph a font substitutes for it is removed. 97

\defaultCJKfontfeatures: Provide \defaultCJKfontfeatures and \addCJKfontfeatures. 74

handle_boundary: A mark keeps its break opportunity beside suspended text: the boundary glue at a mark has no width. . 104

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. 102
LuaCJK.cj_loaded: Provide CJLineBreak, which forbids a line break before the characters whose Unicode Line_Break property is CJ. 85
LuaCJK.define_punct_style: Provide KaiMingPunct, MiddlePunct, LongPunct, and NoBreakLongPunct, each with + and - forms, to edit the punctuation character sets. . . . 92
__LuaCJK_add_fallback:nn: Provide \setCJKfallbackfamilyfont and AutoFallBack for characters that the selected CJK font does not have. 76
__LuaCJK_ignore_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. 79
__LuaCJK_load_fandol:: Provide the LoadFandol option, which supplies Fandol fonts when the document declares none. 80
__LuaCJK_set_autofake:nn: Provide AutoFakeBold, AutoFakeSlant, EmboldenFactor, and SlantFactor; a family that sets BoldFont or ItalicFont is not emboldened or slanted. 52
__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. 43
__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. 58
__LuaCJK_update_font:: Provide PunctFamily, which sets a separate CJK family for the punctuation marks. 69
__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. 62
\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. 49
\LuaCJKDeclarePunctStyle: Provide \LuaCJKDeclarePunctStyle for a punctuation style of your own, with the keys compress-adjacent, adjacent-ratio, and keep-open-close. 51

\LuaCJKDeclareSubCJKBlock: Provide \LuaCJKDeclareSubCJKBlock with \LuaCJKCancelSubCJKBlock and \LuaCJKRestoreSubCJKBlock, so a range of characters can use its own font inside a family. . . . 46
\LuaCJKEditPunctStyle: Provide \LuaCJKEditPunctStyle to edit the punctuation sets of the current style, and AllowBreakBetweenPuncts. . . . 44
\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. 157
\LuaCJKenable: Provide \LuaCJKdisable and \LuaCJKenable to suspend the typographic adjustments while keeping CJK font selection; the state is recorded per node. 59
\LuaCJKnobreak: Provide \LuaCJKnobreak and the PunctWidth option; neither this prohibition nor that of \LuaCJKVerbAddon depends on PunctKinsoku, which controls only the punctuation rules. . . . 59
\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. 59
\LuaCJKsetwidth: Provide \LuaCJKsetwidth and \LuaCJKsetkern for per-mark widths and per-pair gaps that override the selected punctuation style. 45
\LuaCJKShipoutHook: Provide \LuaCJKShipoutHook, which runs code through the standard shipout/before hook. 59
\LuaCJKversion: Provide \LuaCJKversion, which prints the version of the package. . . . 41
\normalspacedchars: Provide \normalspacedchars so that slashes, backslashes, and hyphens take no glue on either side. 78
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. 97
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. 117
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. 103

\youyuan: Provide \songti, \heiti, \fangsong, \kaishu, \lishu, and \youyuan as shorthands for the families zhsong, zhhei, zhfs, zhkai, zhli, and zhyou. 78

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. 110

\CJKfontspec: \CJKfontspec and \newCJKfontfamily take their options after the font name as well. 77

\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. 153

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. 132

__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. 76

__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. 47

__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. 68

\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. 59

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. 131

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. . . . 106

\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. 77