

LuaTeX-ja 和文処理グルーについて

2011/6/27

本文書では、LuaTeX-ja が（現時点において）和文処理に関わる glue/kern をどのように挿入するかの内部処理について説明する。

これは仕様・内部処理の提案の 1 つにしかすぎません。最終的にこのようになる保証はどこにもありませんし、現時点での Lua コードが本文書に従っている保証もありません。バグが混入している可能性も大きいです。

予備知識

説明に入る前に、段落や hbox の中身は、TeX の内部では node 達によるリストとして表現されていることに注意する。node の種類については、*The LuaTeX Reference* の第 8 章を参照して欲しい。代表的なものを挙げると、

- *glyph_node*: 文字（合字も含む）を表現する。和文処理グルーを挿入するには、既に各 *glyph_node* が欧文文字のものか和文文字のものか区別がついている。また、しばしば *glyph_node p* と、その表す文字の文字コード *p.char* とを同一視する。
- *glue_node*: glue を表す。
- *kern_node*: kern を表す。各 *kern_node* には *subtype* という値があり、次の 3 種類を区別できるようになっている。
 - 0: 欧文用 TFM 由来
 - 1: 明示的な `\kern` か、イタリック補正 (\sloppy) によるもの
 - 2: `\accent` による非数式アクセント用文字の左右位置調整のためのもの
- *penalty_node*: penalty を表す。
- *hlist_node*: hbox（水平ボックス）を表す。
- 次のように、node がどのように連続しているかを表すことにする。

$$\boxed{a} \longrightarrow \boxed{b}_1 \longrightarrow \boxed{c}$$

下添字は、LuaTeX-ja においてその node の役割を区別するためにつけられた値（*icflag* と呼ぼう）であり、次のようになっている。

- | | |
|------------------------------------|-----------------------------------|
| 1: イタリック補正由来の kern | 2: 幅補正のため、hbox にカプセル化された和文文字 |
| 3: 禁則処理用 penalty | 4: JFM 由来の glue/kern |
| 5: 「行末」との間に入る kern | 6: <code>\kanjiskip</code> 用 glue |
| 7: <code>\xkanjiskip</code> 用 glue | 8: 既に処理対象となった node |
| 15: リスト先頭/末尾に入る glue/kern/penalty | |

和文処理グルーの挿入処理に一度通された node は、みな *icflag* が 3 以上となることに注意。なお、上添字は node の *subtype* を表す。

- `jaxspmode` のようなサンセリフ体で、`\ltjsetparameter` で設定可能なパラメタ値を表す。
- タイプライタ体の `\kanjiskip`, `\xkanjiskip` は、それぞれ「和文間空白」「和欧文間空白」の意味で抽象的に用いている。
- *nil* 値は \emptyset と書く。

方針など

本バージョンにおいては、JFM 由来グルーと `\[x]kanjiskip` の挿入は同じ段階で行われる。大雑把に言うと、

和文処理グルーの挿入処理では、以下は存在しないものとして扱われる：

- 「文字に付属する」アクセントやイタリック補正。
- 行中数式の内部。
- 実際の組版中には現れない `insertion`, `vadjust`, `mark`, `whatsit node` 達。

和文文字の「自然長」(JFM における `width` の指定値) について

`pTeX` においては、和文文字の行頭と行末に自動的に `glue` や `kern` をおくことはできなかったことから、JFM における文字幅の意味は、

「その文字が行頭におかれるときの、版面左端の位置」を左端、

「その文字が行末におかれるときの、版面右端の位置」を右端としたときの幅

というように、明確な意味があるものであった。例えば、乙部さんによるぶら下げ組パッケージ (`burasage.lzh`) においては、句読点類 (「、、。。」の4文字) の文字幅は 0.0 となっている。

一方、`LuaTeX-ja` においては、和文文字が行末にきた場合、その文字と行末の間に `kern` を挿入することができる：例えば、前に挙げた4文字についてぶら下げ組をしたいのであれば、

```
[1701] = {
  chars = { 'lineend' }
},
[42] = {
  chars = { '、', '、', '、', '、', '。', '。', '。', '。' },
  align = 'left', left = 0.0, down = 0.0,
  width = 0.5, height = 0.88, depth = 0.12, italic=0.0,
  kern = { [1701] = -0.5, ... }
}, ...
```

のように、「文字 `'lineend'` との間に負の `kern` をおく」ように指定すればよい。そのため、`pTeX` と比較すると、JFM における `width` の指定値に絶対的な意味はあまりないことになる。行頭にも `kern` をおけるようにするかどうかは検討中である。

グルーの挿入単位「塊」

和文処理グルーの挿入処理は、ごく大雑把にいうと、「連続する2つの `node` の間に何を挿入するか」の繰り返しである。実際の挿入処理は、「隣り合った2つの『塊』 N_q, N_p の間に何を挿入するか」を単位として行われる。

定義 「塊」(N_n など表す) とは、次の4つのいずれかの条件を満たす `node` (達のリスト) のことである：

1. `icflag` が 3 以上 15 未満である `node` 達の連続からなるリスト。

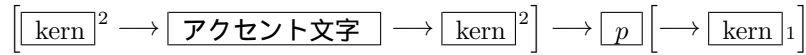
このような `node` 達は、既に組み上がった `hbox` を `\unpackage` により解体したときに発生する。一度和文処理グルーの挿入処理が行われているため、二重の処理を防ぐためにこうして1つの塊を構成させている。

なお、`icflag` が 15 である `node` は、処理中に発見されしだい削除される (`hbox` の先頭や末尾に挿入された `glue/kern/penalty` であるので、本来の「段落/`hbox` の中身に適宜グルーを挿入する」という目的を考えると存在すべきでない)。

2. 数式開始を表す `math_node` から始まる文中数式を表す `node` のリスト：

数式境界 (開始) → (この間、行中数式が続く) → 数式境界 (終了)

3. *glyph_node* p と、それと切り離すことが望ましくないと考えられる node 達：



但し、これには p が *icflag* = 2 の *hbox* である場合も含む。この場合の処理は実はおこらない？

4. 以下のどれにもあてはまらない node p ：

- 組版結果からは消えてしまう, *ins_node*, *mark_node*, *adjust_node*, *whatsit_node* .
- *penalty* (但し, 挿入処理の過程で値が変更されることはある)

記号 B_p で、塊 N_q と塊 N_p の間にある *penalty_node* 達の配列を表す。

挿入処理の大枠

「塊」の保持するデータ

「塊」 N_p は、内部では少なくとも次の要素を持ったテーブルとして表される：

- *.first*: N_p の先頭の node .
- *.nuc*: N_p の「核」となる node .
 - 1., 2. によるものである場合, $N_p.nuc = N_p.first$.
- *.last*: N_p の最後の node .
- *.id*: N_p の種類を表す値 .
 - 1. によるものである場合, *id_pbox* (Pseudo BOX のつもり) .
 - 3. によるものであり, p が和文文字だった場合, *id_jglyph* .
 - 4. によるものであり, p が垂直変位が non-zero な *hbox*, あるいは *vbox*, *rule* だった場合, *id_box_like* .
 - それ以外の場合, node p の種別を表す数値 $p.id$ そのもの . (数値そのものだと使い勝手が悪いので, *id_glyph*, *id_glue*, *id_kern* などと別名を定義している)

定義 「 N_p の中身の先頭」を意味する *head(Np)* は、以下で定義される：

(説明の都合で作った記法で、Lua ソース中にはこのような書き方はない)

- $N_p.id$ が *id_hlist* の場合：後に述べる *check_box* 関数を用いて, *hbox* $N_p.nuc$ 中の「最初の node」「最後の node」を求める .
- $N_p.id$ が *id_pbox* の場合：*id_hlist* の場合とほぼ同様 .
- $N_p.id = id_glyph$ (欧文文字) の場合：
 - *glyph_node* $N_p.nuc$ が単一の文字を格納している (合字でない) 場合は, $N_p.nuc$ 自身 .
 - そうでない場合は, 合字の構成要素の先頭 構成要素の先頭 と再帰的に探索し, 最後にたどり着いた *glyph_node* .
- $N_p.id = id_disc$ (discretionary break) の場合：*disc_node* は,
- $N_p.id = id_jglyph$ (和文文字) の場合： $N_p.nuc$ 自身 .
- $N_p.id = id_math$ (数式境界) の場合：「文字コード -1 の欧文文字」を仮想的に考え, それを *head(Np)* とする .
- それ以外の場合：未定義 . 敢えて書けば $head(Np) := \emptyset$.

同様にして、「 N_p の中身の先頭」を意味する *last(Np)* も定義され、「 N_p は、先頭が *head(Np)*, 末尾が *tail(Np)* であるような単語」のように考えることができる .

定義 「*glyph_node h* の情報を算出する」とは、 $h \neq \emptyset$ の時に、テーブル N_p に以下のような要素を追加することである：

- *.pre*: h の文字コードに対する *prebreakpenalty* パラメタの値
- *.post*: h の文字コードに対する *postbreakpenalty* パラメタの値
- *.xspc_before*, *.xspc_after*: h の前後に `\xkanjiskip` が挿入可能であるかの指定値 (パラメタ *jaxspmode*, *alxspmode* 由来)
- *.auto_xspc*: h での *autoxspacing* パラメタの値

h が和文文字を格納している場合は、さらに次の要素の追加作業も含む：

- *.size*: h で使われている和文フォントのフォントサイズ .
- *.met*, *.var*: 使われている JFM の情報 .
- *.auto_kspc*: *autospadding* パラメタの値 .

全体図

1. 変数類の初期化

- 処理対象が段落の中身 (後で行分割される) の場合: $mode \leftarrow \top$
 - lp (node 走査用カーソル) の初期位置は、リスト先頭部にある `\parindent` 由来の *hbox* や *local paragraph* (Ω 由来) 等の情報を格納する *whatsit node* たちが終わった所 (つまり、段落本来の先頭部分) となる .
 - $last$ (リスト末尾の node) も、リストの最後部に挿入される `\parfillskip` 由来の *glue* を指す .
- 処理対象が *hbox* の中身の場合: $mode \leftarrow \perp$
 - lp はリスト先頭 .
 - 番人として、リスト末尾に *kern* を挿入 . $last$ はこの *kern* となる .

2. 先頭が lp 以降にある塊で、一番早いものを N_p にセットする .

- 作業の途中で $lp = last$ となったら、処理対象のリストに塊はないので、8. へ .
- そうでなければ、 $head(N_p)$ の情報を算出しておく .
- 本段階終了後、 lp は $N_p.last$ の次の node となる .

3. (*handle_list_head*) リストに最初に出てくる塊 N_p が求まったので、リスト「先頭」とこの塊との間に和文処理グルーを挿入 .

4. 今の塊 N_p と、その次の塊の間に入る和文処理グルーを求めるため、一旦 $N_q \leftarrow N_p$ として待避させ、次の塊 N_p を探索する .

- 作業の途中で $lp = last$ となったら、 N_q がリスト中最後の塊であるので、7. へ .
- そうでなければ、 $head(N_p)$ の情報を算出しておく .
- 本段階終了後、 lp は $N_p.last$ の次の node となる .

5. N_q と N_p の間に和文処理グルーを挿入する . $N_p.id$ による場合分けを行う . 「main loop その 1, 2」を参照のこと .

6. N_p が単一の文字ではない (合字など) 可能性がある以下の場合において、 $tail(N_p)$ の情報を算出する . 終わったら、再びループに入るため、4. へ .

- *id_glyph* (欧文文字) のとき
- *id_disc* (discretionary break) のとき
- *id_hlist* のとき
- *id_pbox* のとき

7. (`handle_list_tail`) リストの最後にある塊 N_q が求まったので、この塊とリスト「末尾」の間に和文処理グルーを挿入。
8. $mode = \perp$ の場合、番人となる kern を 1. において挿入したので、その番人を削除する。

リスト先頭・末尾の処理と「box の内容」

リスト先頭の処理 (`handle_list_head`)

次の場合に、 N_p で使われているのと同じ JFM を使った「文字コードが `'boxbdd'` の文字」と N_p との間に入る glue/kern を、 $N_p.first$ の直前に挿入する：

- $N_p.id = id_jglyph$ (和文文字)
- $N_p.id = id_pbox$ であり、 $head(N_p)$ が和文文字であるとき。

$mode = \perp$: $(\text{リスト先頭}) \rightarrow \dots \rightarrow \boxed{g}_{15} \rightarrow \boxed{N_p}$
 $mode = \top$: $\boxed{\backslash parindent \text{ 由来 hbox}} \rightarrow \dots \left[\rightarrow \boxed{\text{penalty } 10000}_{15} \right] \rightarrow \boxed{g}_{15} \rightarrow \boxed{N_p}$

ここで、 g が glue かつ $mode = \top$ かつ $\#Bp = 0$ のときのみ、`\parindent` 由来の `hbox` の直後で改行されることを防ぐために g の直前に `penalty` を挿入する。($\#Bp$ が 1 以上の場合は、`\parindent` と N_p の間にある `penalty` のため、 N_p の直前での改行が起こり得る状態となっているので、特にそれを抑制することもしない)。

リスト末尾の処理 (`handle_list_tail`)

この場合、 $mode$ の値により処理が全く異なる。

A: $mode$ が偽である場合。

この場合はリストは `hbox` の中身だから、行分割はおこり得ない。リスト先頭の処理と同様に、次の場合に N_q と「文字コードが `'boxbdd'` の文字」との間に入る glue/kern を、 $N_q.last$ の直後に挿入する：

- $N_q.id = id_jglyph$ (和文文字)
- $N_q.id = id_pbox$ であり、 $tail(N_q)$ が和文文字であるとき。

$\boxed{N_q} \rightarrow \boxed{g}_{15} \rightarrow \dots \rightarrow \boxed{\text{kern(番人)}}$

上の番人は、次の step で除去されるのだった。

B: $mode$ が真である場合。

この場合、段落の末尾には常に `\penalty 10000` と `\parfillskip` 由来のグルーが存在する。そのため、上のように「文字コードが `'boxbdd'` の文字」との空白を考えるのではなく、まず、 N_q が行末にきたときに行末との間に入る空白 w を代わりに挿入する。

- $N_q.id = id_jglyph$ (和文文字)
- $N_q.id = id_pbox$ であり、 $tail(N_q)$ が和文文字であるとき。

$\boxed{N_q} \rightarrow \boxed{\text{kern } w}_{15} \rightarrow \boxed{\text{penalty } 10000} \rightarrow \dots \rightarrow \boxed{\text{glue}(\backslash\text{parfillskip})}$

次に、`\jcharwidowpenalty` の挿入処理を行う 省略。

box 内の「最初/最後の文字」の検索 (check_box)

「hbox 中の文字と外の文字の間に」\kanjiskip, \xkanjiskip の挿入を行えるようにするため, check_box 関数では hbox 内の「最初の node」「最後の node」の検索を行う.

- 以下の node は検索から除外される:
 - 組版結果からは消えてしまう *ins_node*, *mark_node*, *adjust_node*, *whatsit_node*, *penalty* .
 - (box 中身の先頭/末尾に入っている) *icflag* が 7 の *glue/kern/penalty* .
 - アクセント部とイタリック補正 .
- *hlist_node* *q* に出会ったら, *q* の垂直変位量が 0 である限り, 検索は *q* の内部も進む. 以下同文 .
- 検索して得られた「最初の node」「最後の node」がそれぞれ *glyph_node* でなければ, 実際には \emptyset を返す .

main loop

一覧表

Nq , Np の種類別に挿入される glue/kern の種別を表にすると次のようになる。

Nq Np	和文 1	和文 2	欧文	箱	id_glue	id_kern
和文 1	E M K nor	× O _A K nor	— O _A X nor	— all	— O _A nor	— O _A sup
和文 2	E O _B K nor ×	× sup ×	— sup X sup ×			
欧文	E O _B X nor	— sup X sup ×				
箱	E O _B alw —					
id_glue	E O _B nor —					
id_kern	E O _B sup —					

- 項目名 表 1 行目の Nq の種類について説明する。 Np についても同様。
 - 「和文 1」: リスト中に直接出現している和文文字。
 - $Nq.id = id.jglyph$ であったとき。
 - $Nq.id = id.pbox$ かつ $last(Nq)$ が和文文字であったとき。
 - 「和文 2」: リスト内にある hbox の中身として出現した和文文字。すなわち, $Nq.id = id.hlist$ かつ $last(Nq)$ が和文文字であったとき。
 - 「欧文」: $last(Nq)$ が欧文文字であったとき。即ち,
 - リスト中に直接出現しているとき ($Nq.id = id.jglyph$ or $Nq.id = id.pbox$ かつ $last(Nq)$ が欧文文字)。
 - $Nq.id = id.hlist$ かつ $last(Nq)$ が欧文文字であったとき。
 - $Nq.id = id.math$ であったとき。
 - 「箱」: 前後に和文処理グルーが挿入されない用な box 状の node。
 - $Nq.id = id.list$ かつ $last(Nq)$ が文字でなかった (未定義) だったとき。
 - $Nq.id = id.box.like$ のとき。
 - 「id_glue」: そのまま, $Nq.id = id_glue$ であったとき。
 - 「id_kern」: そのまま, $Nq.id = id_kern$ であったとき。

- 表中の各セルは, それぞれ次のような内容を表している:

左空白	右空白
L	P 取扱 R

- 「左空白」: Nq の直後に挿入される空白の種類。空欄は, 何も入らないことを表す。
- 「右空白」: Np の直前に挿入される空白の種類。

なお, 「A B」は, まず A の種類の glue/kern を調べ, それが未定義ならば, B の種類の glue/kern を採用することを示している。このとき, 矢印の右側に入る空白 (K, X) はいつでも定義されていることに注意。
- 「P 取扱」: Nq と Np の間に入る禁則用ペナルティの取扱の方法を表す。 Nq と Np の間で常に行分割を許すかに伴い, normal, always, suppress の 3 種類がある。
- 「L」「R」: 禁則用ペナルティの挿入処理において, $Nq.post$ (L) や $Np.pre$ (R) の値を実際に活用するかどうかを示す。値は次の 3 種類:

(利用する), × (利用せず, 0 として扱う), — (未定義のため 0 扱い)

挿入される glue/kern の種類

前節の表にある空白の種類についての解説を行う。

- E: Nq が行末にきたとき, Nq と行末の間に入る空白 (kern). 挿入位置は $Nq.last$ の直後.
 - JFM では「文字コード 'lineend' の文字」との間に入る kern 量として設定できる.
 - 右空白が kern であるときは挿入されない.
 - この種類の kern が挿入される時, 右空白は自然長が E の分だけ引かれる.

- M: Nq と Np の間に入る JFM 由来の glue/kern.
 - Nq, Np の間で `\inhibitglue` を発行した場合, 挿入は抑止される.
 - 両方の塊で使われている JFM が (サイズもこめて) 等しい場合は, 両者で使われている JFM の情報をそのまま利用できるので, 量の決定は容易い.
 - そうでなければ, まず
$$gb := (Nq \text{ と「文字コードが 'diffmet' の文字」の間に入る glue/kern})$$
$$ga := (\text{「文字コードが 'diffmet' の文字」} \text{ と } Np \text{ との間に入る glue/kern})$$
として 2 つの量を計算. 少なくとも片方が未定義の場合は, もう片方の値を用いる. そうでなければ, 両者の値から自然長, 伸び量, 縮み量ごとに計算 (方法として, 平均, 和, 大きい方, 小さい方) を行い, それによって得られた glue/kern を採用する.

- K: `\kanjiskip` を表す glue を挿入 (\emptyset にはならない).
 - 両方の塊において「`\kanjiskip` の自動挿入が無効」($Nq.auto_kspc \vee Np.auto_kspc = \perp$) ならば, 長さ 0 の glue を挿入する.
 - `\kanjiskip` パラメタの自然長が $\maxdimen = (2^{30} - 1) \text{ sp}$ であれば, JFM に指定されている `\kanjiskip` の量を用いる. Nq, Np で使われている JFM が異なった時の処理は, M の場合と同じである.
 - 上のどれにも当てはまらなければ, `\kanjiskip` パラメタで表される量の glue を挿入する.

- X: `\xkanjiskip` を表す glue を挿入 (\emptyset にはならない).
 - 次のいずれかの場合には, `\xkanjiskip` は長さ 0 の glue となる:
 - 両方の塊において, 「`\xkanjiskip` の自動挿入が無効」という指定 ($Nq.auto_xspc \vee Np.auto_xspc = \perp$) がされていた場合.
 - Nq 内の文字について「直後への `\xkanjiskip` 挿入が無効」であった場合, 即ち $alxspmode \geq 2$ (欧文) か $jaxspmode \equiv 0 \pmod{2}$ (和文).
 - Np 内の文字について「直前への `\xkanjiskip` 挿入が無効」であった場合, 即ち $alxspmode \equiv 0 \pmod{2}$ (欧文) か $jaxspmode \geq 2$ (和文).
 - `\xkanjiskip` パラメタの自然長が \maxdimen であれば, $last(Nq), head(Np)$ の片方が和文文字であるので, そこで使われている JFM で指定されている `\xkanjiskip` の量を用いる (JFM で指定されていなければ長さ 0 の glue と見なされる).
 - 上のどれにも当てはまらなければ, `\xkanjiskip` パラメタで表される量の glue を挿入する.

- O_B: Nq と「文字コードが 'jcharbdd' の文字」との間に入る glue. M のバリエーションと考えればよく, 同じように `\inhibitglue` の指定で抑止される.

- O_A: 「文字コードが 'jcharbdd' の文字」と Np との間に入る glue. M のバリエーションと考えればよく, 同じように `\inhibitglue` の指定で抑止される.

penalty まわりの処理

隣り合った塊 Nq , Np の間には, 集合 Bp で表される 0 個以上の penalty があるのだった:

$$\boxed{Nq} \left[\rightarrow \boxed{E} \right]_4 \rightarrow \cdots (\text{penalty ある可能性あり}) \cdots \left[\rightarrow \boxed{M, K, X \text{ 他}} \right]_{3, 5, 6} \rightarrow \boxed{Np}$$

禁則処理に関する penalty の挿入処理は, 以下に述べるところ部分は共通の動作である.

$Bp \geq 1$ の場合には, 全ての Bp の元 p (penalty) に対して次を行う:

$$p.\text{penalty} += a, \quad a := Nq.\text{post} + Np.\text{pre}.$$

- 全ての Bp の元に対して行うのは, 実際にはどの penalty の位置で行分割が行われるかわからないからである.
- 数ページ前の表で, 左下が「×」or「—」となっていた場合は, 上の計算式において $Nq.\text{post}$ は 0 と扱われる. 右下が「×」or「—」なら, $Np.\text{pre}$ が 0 と扱われる.
- penalty 値の計算では, +10000 は正の無限大, -10000 は負の無限大として扱っている. そのため, a の計算や $p.\text{penalty}$ への加算代入のところでは, 通常に加減算で絶対値が 10000 を越えたら分はカットし, さらに $(10000) + (-10000) = 0$ としている.

$Bp = 0$ の場合が, penalty 挿入の 3 種類の方法「normal」「always」「suppress」で異なる部分である:

- 「normal」の場合: 次の場合に, $p.\text{penalty} = a$ である penalty p を作成し, それを (M, K 他の glue 挿入前に) $Np.\text{first}$ の直前に挿入する:

左空白 (E) が存在しているか, $a \neq 0$ かつ右空白が kern である.

- 「always」の場合: この場合は, Nq , Np の間で常に行分割可能にしたいので, 挿入する条件は以下ようになる:

左空白 (E) が存在しているか, 右空白が glue でない (つまり, kern が未定義のとき).

- 「suppress」の場合: このとき, Nq と Np の間での行分割は元々不可能である. Lua_{TeX}-ja では, そのような場合を「わざわざ行分割可能に」することはしない. つまり, 右空白が glue であるとき, その直前に `\penalty 10000` を挿入する.

いくつかの例: 未完

main loop その 2: その他の場合

Np Nq	id_hlist 非文字			
	$head(Nq)$: 欧文	id_box_like	id_glue	id_kern
id_jglyph	$E + (O_B \quad X)$	$E + O_B^*$	$E + O_B$	$E + O_B^+$
id_pbox 和	$E + (O_B \quad X)$	$E + O_B^*$	$E + O_B$	$E + O_B^+$
id_hlist 和	X^+	—	—	—
他	—	—	—	—