LuaTEX-ja 和文処理グルーについて

2011/6/18

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，これを書いている時点では実際の **Lua** コードは書きあがっていません．

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際には，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，それ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 (`\/`) 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: 禁則処理用 penalty
  3: JFM 由来の glue/kern
  4:「行末」との間に入る kern
  5: `\kanjiskip`用glue
  6: `\xkanjiskip`用glue
  7: リスト先頭/末尾に入る glue/kern/penalty
  8: 既に処理対象となった node

  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 2 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』Nq, Np の間に何を入れるか」を単位として行われる．

定義　「塊」（Nn などで表す）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 2 以上 6 以下 or 8 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 7 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：

$$\left[\boxed{\text{kern}}^{2} \longrightarrow \boxed{\text{アクセント文字}} \longrightarrow \boxed{\text{kern}}^{2}\right] \longrightarrow \boxed{p}\left[\longrightarrow \boxed{\text{kern}}_{1}\right]$$

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　Bp で，塊 Nq と塊 Np 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」Np 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*: Np の先頭の node.
- *.nuc*: Np の「核」となる node.
  – 1., 2. によるものである場合，$Np.nuc = Np.first$.
- *.last*: Np の最後の node.
- *.id*: Np 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*.
  – 3. 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　「Np の中身の先頭」を意味する $head(Np)$ は，以下で定義される：
（説明の都合上作った記法で，Lua ソース中にはこのような書き方はない）

- $Np.id$ が *id_hlist* の場合：後に述べる `check_box` 関数を用いて，hbox $Np.nuc$ 中の「最初の node」「最後の node」を求める.
- $Np.id$ が *id_pbox* の場合：*id_hlist* の場合とほぼ同様.
- $Np.id = id\_glyph$（欧文文字）の場合：
  – *glyph_node* $Np.nuc$ が単一の文字を格納している（合字でない）場合は，$Np.nuc$ 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*.
- $Np.id = id\_disc$ (discretionary break) の場合：*disc_node* は，
- $Np.id = id\_jglyph$（和文文字）の場合：$Np.nuc$ 自身.
- $Np.id = id\_math$（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，「Np の中身の先頭」を意味する $last(Np)$ も定義され，「Np 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 Np 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*: autospacing 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：$mode \leftarrow \bot$
     - 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 Np 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，$head(Np)$ の情報を算出しておく．
   – 本段階終了後，lp は $Np.last$ の次の node となる．
3. (`handle_list_head`) リストに最初に出てくる塊 Np 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 Np 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 $Nq \leftarrow Np$ として待避させ，次の塊 Np を探索する．
   – 作業の途中で $lp = last$ となったら，Nq がリスト中最後の塊であるので，7. へ．
   – そうでなければ，$head(Np)$ の情報を算出しておく．
   – 本段階終了後，lp は $Np.last$ の次の node となる．
5. Nq と Np の間に和文処理グルーを挿入する．$Np.id$ による場合分けを行う．「main loop その 1, 2」を参照のこと．
6. Np 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，$tail(Np)$ 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 Nq 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. $mode = \bot$ 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，Np 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 Np との間に入る glue/kern を，$Np.first$ の直前に挿入する：

- $Np.id = id\_jglyph$（和文文字）
- $Np.id = id\_pbox$ であり，$head(Np)$ が和文文字であるとき．

$mode = \bot$: $\big|$（リスト先頭）$\longrightarrow \cdots \longrightarrow \boxed{g}_7 \longrightarrow \boxed{Np}$

$mode = \top$: $\boxed{\texttt{\textbackslash parindent} 由来 \text{hbox}} \longrightarrow \cdots \Big[\longrightarrow \boxed{\text{penalty } 10000}_7\Big] \longrightarrow \boxed{g}_7 \longrightarrow \boxed{Np}$

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 Np の間にある penalty のため，Np 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 Nq と「文字コードが 'boxbdd' の文字」との間に入る glue/kern を，$Nq.last$ の直後に挿入する：

- $Nq.id = id\_jglyph$（和文文字）
- $Nq.id = id\_pbox$ であり，$tail(Nq)$ が和文文字であるとき．

$$\boxed{Nq} \longrightarrow \boxed{g}_7 \longrightarrow \cdots \longrightarrow \boxed{\text{kern（番人）}}$$

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，Nq 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．

- $Nq.id = id\_jglyph$（和文文字）
- $Nq.id = id\_pbox$ であり，$tail(Nq)$ が和文文字であるとき．

$$\boxed{Nq} \longrightarrow \boxed{\text{kern } w}_7 \longrightarrow \boxed{\text{penalty } 10000} \longrightarrow \cdots \longrightarrow \boxed{\text{glue (\textbackslash parfillskip)}}$$

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 の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 その 1: $head(Np)$ が和文文字の場合

これは，次の 3 つの場合でおこる：

- $Np.id$ が *id_jglyph* の場合（本当に和文文字）
- $Np.id$ が *id_pbox* で，$head(Np)$ が和文文字の場合
- $Np.id$ が *id_hlist* で，$head(Np)$ が和文文字の場合

前 2 つの場合は，$head(Np)$ は処理対象のリスト中に現れる本当の *glyph_node* である．一方，最後の場合では，$head(Np)$ はリスト中にある hbox の中身（の最初）に出現する *glyph_node* である．

そのため，挿入される和文処理グルーの種類については，前 2 つと最後の場合とで扱いを異なったものとしている：

> **hbox** の外側の文字と内側の文字の間の空白では，`\kanjiskip`, `\xkanjiskip` の量の計算では両方の文字の情報を使うが，**JFM** 由来のグルーでは内側の文字の情報は使われない．

■挿入される **glue/kern** の種類

$Nq \rightarrow$ / $Np \downarrow$	*id_jglyph* / *id_pbox* 和	*id_hlist* 和	$head(Nq)$: 欧文	*id_hlist* 非文字 *id_box_like*	*id_glue*	*id_kern*
id_jglyph	$\mathrm{E} + (\mathrm{M} \rightarrow \mathrm{K})$	$\mathrm{O_A} \rightarrow \mathrm{K}$	$\mathrm{O_A} \rightarrow \mathrm{X}$	$\mathrm{O_A^*}$	$\mathrm{O_A}$	$\mathrm{O_A^+}$
id_pbox 和	$\mathrm{E} + (\mathrm{M} \rightarrow \mathrm{K})$	$\mathrm{O_A} \rightarrow \mathrm{K}$	$\mathrm{O_A} \rightarrow \mathrm{X}$	$\mathrm{O_A^*}$	$\mathrm{O_A}$	$\mathrm{O_A^+}$
id_hlist 和	$\mathrm{E} + (\mathrm{O_B} \rightarrow \mathrm{K})$	$\mathrm{K^+}$	$\mathrm{X^+}$	—	—	—

挿入される glue/kern の種別を表にすると上のようになる．最後の 1 つ以外は，挿入される位置は *Np.first* の直前であり，以降「右側の空白」と呼ぶ．

- 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）等しかったら量の決定は容易い．
  - そうでなければ，まず

    $gb :=$ (Nqと「文字コードが`'diffmet'`の文字」との間に入るglue/kern.)

    $ga :=$ (「文字コードが`'diffmet'`の文字」とNp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 $\emptyset$ 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」($Nq.auto\_kspc \vee Np.auto\_kspc = \bot$) ならば，長さ 0 の glue を挿入する．
  - kanjiskip パラメタの自然長が `\maxdimen` $= (2^{30} - 1)\,\mathrm{sp}$ 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）．
  - 両方の塊において「`\xkanjiskip` の自動挿入が無効」($Nq.auto\_xspc \vee Np.auto\_xspc = \bot$) ならば，長さ 0 の glue を挿入する．

– Nq内の文字が「直後への \xkanjiskip 挿入が無効」という指定 (alxspmode ≥ 2) であるか，Np内の文字が「直前への \xkanjiskip 挿入が無効」という指定 (jaxspmode ≥ 2) であるならば，長さ 0 の glue を挿入する．
– xkanjiskip パラメタの自然長が \maxdimen であれば，JFM に指定されている \xkanjiskip の量を用いる．
– 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．

- $\mathrm{O_B}$: 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．
- $\mathrm{O_A}$: 「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．
- 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．
  – 右側の空白が kern であるときは挿入されない．
  – この種の kern が挿入される時，右側の空白は自然長が E の分だけ引かれる．

あと，注として，

- 「→」は，左側の種類（例えば M）の glue/kern は $\emptyset$ であった場合，右側の種類（例えば K）の glue を挿入することを示す．
- *, $^+$ は，penalty 処理時のバリエーションを表す．次の節では，上添字なしの場合の penalty の処理について述べる．

■ **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：

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

このような状況下で，禁則処理に関係する penalty の挿入処理は，原則として（上ほど優先度高）：

- $\#Bp \geq 1$ の場合，全ての Bp の元 p (penalty) に対して，

$$p.penalty \mathrel{+}= Nq.post + Np.pre.$$

  – 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．
  – $Nq.id = id\_hlist$ の場合には，$Nq.post$ は 0 と扱われる．$Np.id = id\_hlist$ の場合も同様．

- $\#Bp = 0$ かつ $Nq.post + Np.pre =: a \neq 0$，さらに「右側の空白が kern でない」場合：$p.penalty = a$ である penalty p を作成し，それを（M, K 他の glue 挿入前に）$Np.first$ の直前に挿入する．つまり，この場合，

$$\longrightarrow \cdots \longrightarrow \boxed{\text{penalty } a}\left[\longrightarrow \boxed{\text{M, K, X 他}}_{3,\,5,\,6}\right] \longrightarrow \boxed{Np}$$

  となる．

- $\#Bp = 0$ かつ $\mathrm{E} \neq 0$（かつ右側の空白が glue）の場合：同様に新たな penalty を作る．
  (E の位置で改行可能にしたいので)

- つまり，$\#Bp = 0$ であったとき，新たな penalty を作らないのは，$\mathrm{E} = 0$ かつ $a = 0$ の場合に限る．

なお，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，通常の加減算で絶対値が 10000 を越えたら，その分はカットしている．あと，

$(10000) + (-10000) = 0$ としている．また，$Nq.post$, $Np.pre$ が $\emptyset$ の場合は，それぞれ 0 として扱う．

バリエーションについて　前節の表に出てきた *, $^+$ では，上の原則から以下の点が変更されている．変更点は，いずれも $\#Bp = 0$ の場合に関するところのみである：

- *: Nq と Np の間での行分割を常に可能とするため，右側の空白（$\mathrm{O_A}$）が $\emptyset$ の場合であっても新たな penalty を作る．
- $^+$: このとき，Nq と Np の間での行分割は元々不可能である．LuaTEX-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．そのため，
  - 右側の空白が glue の場合は，値が 10000 の penalty を作成する．
  - 右側の空白が $\emptyset$ か kern の場合は，新たに penalty を作ることはしない．

■いくつかの例：未完

main loop その 2: その他の場合

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