ruby パッケージ

Naoki Kaneko (a.k.a. "puripuri2100")

目次

1. 概要

SATYSFI でルビを使うためのパッケージです。モノルビ・グループルビ・熟語ルビに対応し、JLReq になるべく準拠するようにしています。

現在のバージョンは 0.1.2 です。

2. 基本的な使い方

インストールは基本的に satyrographos で行われることを想定しています。 手動で行う場合は、適切な位置に配置し、読み込みも適切に行ってください。

ruby ライブラリによって提供されるパッケージは ruby のみです。 そのため、 読み込みはこのようにして行います。

```
@require: ruby/ruby
```

提供されるコマンドは `\ruby` のみです。 コマンドと同時に関連する関数も提供されます。 モジュール名は `Ruby` です。

`\ruby` コマンドは引数を 3 つ取ります。

(1) 設定（オプション引数）

(2) ルビ文字（string list）

(3) 親文字（inline-text list）

例えば、`\ruby[` こ `; ` ばと `]{| 小 | 鳩 |}` のように入力すると「小鳩（こばと）」と出力されます。

3. ルビのモード

3.1. モードの切り替え

デフォルトではモノルビが選択されますが、熟字訓などを使いたい場合はグループルビを、熟語にルビを付けたい場合には熟語ルビを、使いたくなると思います。

その場合はオプション引数を使って設定します。 オプション引数は `\ruby ?:[<` 設定関数 `>][` こ `; ` ばと `]{| 小 | 鳩 |}` の形で用います。 例えば、`\ruby ?:[Ruby.mode Ruby.g][` しぐれ `]{| 時 | 雨 |}` で 「時雨（しぐれ）」 となります。`Ruby.mode` は引数として受け取ったモード関数が指定するモードに切り替えるための関数です。`Ruby.g` がグループルビモードを指定する関数です。 他にはモノルビを指定する `Ruby.m`、 熟語ルビを指定する `Ruby.j` が存在します。

3.2. モノルビモードの使い方

現時点のデフォルトモードです。

最初にルビ文字を string list の形で与えます。一つの漢字に付き一つの文字列、というように入力してください。次に親文字を inline-text list の形で与えます。漢字ごとにバラバラにして入力してください。

通常はルビ文字のリスト数と親文字のリスト数は一致しているはずです。ルビ文字が少ない場合、その分は「ルビがない」として扱われ、ルビ文字が多い場合は、溢れたルビは無視されます。

ルビ文字が親文字より短い場合、ルビ文字は中央ぞろえで出力されます。

ルビ文字が親文字より長い場合、親文字は中央ぞろえで出力されます。

以下にいくつか例をあげておきます。

- `\ruby ?:[Ruby.mode Ruby.m] [` こ `;` とり `]{| 小 | 鳥 |}:`「小鳥(ことり)」
- `\ruby ?:[Ruby.mode Ruby.m] [` うぐいす `;]{| 鶯 |}:`「鶯(うぐいす)」
- `\ruby ?:[Ruby.mode Ruby.m] [` しち `;` めん `;` ちょう `;]{| 七 | 面 | 鳥 |}:`「七面鳥(しちめんちょう)」

3.3. グループルビモードの使い方

熟字訓など、分割することのできないルビを振るときに使います。

使い方はモノルビと似ていますが、モノルビと違い、ルビ文字のリスト数は気にする必要がありません。最終的に全て結合された状態で評価されます。親文字の inline-text はしっかり漢字一文字ずつに分割してください。間にスペースを入れる際に不揃いになります。

ルビ文字が親文字より短い場合、ルビ文字は中央ぞろえで出力されます。また、ルビ間に適切な量のスペースが入ります。

ルビ文字が親文字より長い場合、親文字は中央ぞろえで出力されます。また、親文字間に適切な量のスペースが入ります。

以下にいくつか例をあげておきます。

- `\ruby ?:[Ruby.mode Ruby.g] [` こ `;` とり `]{| 小 | 鳥 |}:`「小鳥(ことり)」

- \ruby ?:[Ruby.mode Ruby.g] [` うぐいす `;]{| 鶯 |}:「鶯(うぐいす)」
- \ruby ?:[Ruby.mode Ruby.g] [` しち `;` めん `;` ちょう `;]{| 七 | 面 | 鳥 |}:「七面鳥(しちめんちょう)」
- \ruby ?:[Ruby.mode Ruby.g] [` ライセンス `;]{| 利 | 用 | 規 | 約 |}:「利用規約(ライセンス)」
- \ruby ?:[Ruby.mode Ruby.g] [` クライアント `;]{| 顧 | 客 |}:「顧客(クライアント)」

3.4. 熟語ルビモードの使い方

一かたまりの熟語の親文字間に不要なスペースが入るのを防ぐために文字の位置を融通しあうモードです。

使い方はモノルビと完全に一致します。

JLReq 独自の組み方が複雑であったため、もう一つの選択肢として書かれていた JIS X 4051 の組み方に従っています。

ルビ文字が親文字より長い組み合わせが一つでも存在する場合、グループルビと同じ挙動をします。

ルビ文字が全て親文字より短い場合、全ての組み合わせに対してモノルビが行われたのと同じ挙動をします。

以下にいくつか例をあげておきます。

- \ruby ?:[Ruby.mode Ruby.j] [` こ `;` とり `]{| 小 | 鳥 |}:「小鳥(ことり)」
- \ruby ?:[Ruby.mode Ruby.j] [` うぐいす `;]{| 鶯 |}:「鶯(うぐいす)」
- \ruby ?:[Ruby.mode Ruby.j] [` しち `;` めん `;` ちょう `;]{| 七 | 面 | 鳥 |}:「七面鳥(しちめんちょう)」

4. 進入の調節

「これが鶯(うぐいす)という鳥です。」というような文があったとき、ルビの「う」と「す」の下に前後のひらがなを入れられればスペースが詰まり、読みやすくなると感じる人もいると思います。その場合に進入の調節を行います。

通常、前（もしくは後ろ）のひらがなにはルビを 1 文字文かけることが許可されており、漢字の場合にはルビを 0.5 文字分かけることが許されています。SATySFI で前後の文字の判定や大きさの判定を行うことは難しいため、組版者が手動で進入の大きさの調節を行います。

第一引数の設定部分で Ruby.before-intrusion Ruby.big とすることで、ルビが 1 文字分前にかかります。Ruby.before-intrusion を Ruby.after-intrusion にすることで前後を切り替えられます。両方を同時に指定することも可能です。

- これが \ruby [` うぐいす `]{| 鶯 |} という鳥です。：「鶯（うぐいす）という鳥です。」
- これが \ruby ?:[Ruby.before-intrusion Ruby.big] [` うぐいす `]{| 鶯 |} という鳥です：「これが鶯（うぐいす）という鳥です。」
- これが \ruby ?:[Ruby.after-intrusion Ruby.big] [` うぐいす `]{| 鶯 |} という鳥です：「これが（うぐいす）鶯という鳥です。」
- これが \ruby ?:[Ruby.before-intrusion Ruby.big; Ruby.after-intrusion Ruby.big] [` うぐいす `]{| 鶯 |} という鳥です：「これが鶯（うぐいす）という鳥です。」

上の例で前後の進入についての設定が分ったでしょうか？ Ruby.big では 1 文字分でした。Ruby.small では 0.5 文字分になっています。 デフォルトでは前後共に Ruby.none が指定されています。これは「進入は行わない」という設定です。

5. 突出禁止

行頭や行末でルビのはみ出しを完全に禁止したい場合が存在します。この時は Ruby.small や Ruby.big の代わりに Ruby.suppresses を用いることで実現できます。

その性質上、前後両方に Ruby.suppresses を指定することはできず、モノルビの場合は前側のみ突出禁止が効きます。しかし、グループルビの時は挙動が代わります。グループルビでは通常、 端にも少しスペースを作ることにしていますが、Ruby.suppresses を前後両方にかけることでルビと親文字の端を完全に揃えることができるようになっています。

- \ruby ?:[Ruby.mode Ruby.g] [` ライセンス `;]{| 利 | 用 | 規 | 約 |}：「利用規約（ライセンス）」
- \ruby ?:[Ruby.before-intrusion Ruby.suppresses; Ruby.mode Ruby.g; Ruby.after-intrusion Ruby.suppresses] [` ライセンス `;]{| 利 | 用 | 規 | 約 |}：「利用規約（ライセンス）」
- \ruby ?:[Ruby.mode Ruby.g] [` クライアント `;]{| 顧 | 客 |}：「顧客（クライアント）」

- `\ruby ?:[Ruby.before-intrusion Ruby.suppresses; Ruby.mode Ruby.g; Ruby.after-intrusion Ruby.suppresses] [` クライアント `;]{| 顧 | 客 |}:「顧(クライアント)客」`

6. バグ報告・修正や機能追加の提案

このパッケージはバグが存在するかもしれません。バグを発見した場合は以下の URL に報告してください。

`https://github.com/puripuri2100/SATySFi-ruby/issues`

このパッケージに対してコードの修正や機能追加の提案をしたい場合は、GitHub の機能を用いて以下の URL にプルリクエストを送ってください。

`https://github.com/puripuri2100/SATySFi-ruby/pulls`

バグ報告・修正提案・機能追加提案をお待ちしております。

7. 変更履歴

- 2020/9/30 v0.1.2 ルビとボディの間にスペースを入れる処理を付け加えた
- 2020/9/28 v0.1.1 ドキュメント追加
- 2020/9/27 v0.1.0 作成

8. ライセンスとコピーライト