Le paquet impnattypo

Raphaël Pinson
raphink@gmail.com

1.0 en date du 2011/09/20

1 Introduction

En matière de typographie française, le *Lexique des règles typographiques en usage à l'Imprimerie Nationale* est une référence incontournable.

Si la majorité des recommendations de cet ouvrage est implémentée dans le module frenchb pour babel, certaines autres recommendations méritent encore d'être automatisées pour être implémentées en LaTeX.

C'est le but original de ce paquet, initié par une question sur le site tex.stackexchange.com [1], et qui implémente plusieurs règles édictées dans ce lexique afin de les rendre plus facilement applicables aux textes édités avec LaTeX.

Au fur et à mesure que ce paquet a grandi, des fonctionalités sont venues s'ajouter, dont certaines ne sont pas directement liées au lexique, mais améliorent la qualité typographique des documents.

2 Utilisation

Pour utiliser le paquet `impnattypo`, entrez la ligne:

```
\usepackage[<options>]{impnattypo}
```

Les options du paquet sont décrites dans les sections suivantes.

2.1 Césures

`hyphenation`

En dehors des règles générales de coupure des mots, le lexique indique qu'il faut « [éviter] les coupures de mots sur plus de trois lignes consécutives. »

Afin de simplifier le code, l'implémentation proposée décourage fortement les césures en fin de page, ainsi que les césures sur deux lignes consécutives.

Pour activer cette fonctionalité, utilisez l'option `hyphenation`:

```
\usepackage[hyphenation]{impnattypo}
```

1. `http://tex.stackexchange.com/questions/20493/french-typography-recommendations`

2.2 Formatage des paragraphes

`parindent` Le lexique conseille une indentation des paragraphes de 1em. Ce réglage de `\parindent` peut être obtenu par l'utilisation de l'option `parindent`:

```
\usepackage[parindent]{impnattypo}
```

`lastparline` De plus, il est indiqué dans la section « Coupure des mots » que « la dernière ligne d'un alinéa doit comporter un mot ou une fin de mot de longueur au moins égale au double du renfoncement de l'alinéa suivant. » À défaut d'implémenter exactement cette solution, l'option `lastparline` s'assure que la dernière ligne d'un alinéa est au moins aussi longue que le double de la valeur de `\parindent`. [2]

Lorsque LuaTEXest utilisé, la solution de Patrick Gundlach [3] est utilisée. Avec les autres moteurs de rendu, c'est la solution native de Enrico Gregorio [4] qui fait office d'implémentation:

```
\usepackage[lastparline]{impnattypo}
```

Lorsque l'option `draft` est activée et que LuaTEXest utilisé, les espaces insécables insérés sont colorés en teal. La couleur utilisée peut être ajustée par l'option `lastparlinecolor`.

`nosingleletter` Il est également recommendé d'éviter les coupures isolant une lettre. La solution proposée par Patrick Gundlach [5] permet de remédier à cela en utilisant LuaTEX. Pour activer cette fonctionalité, il faut utiliser l'option `nosingleletter`:

```
\usepackage[nosingleletter]{impnattypo}
```

Lorsque cette option est activée, seul LuaTEX (via la commande `lualatex`) pourra effectuer le rendu du document.

Lorsque l'option `draft` est activée, les espaces insécables insérés sont colorés en brown. La couleur utilisée peut être ajustée par l'option `nosingleletterecolor`.

`homeoarchy` Lorsque deux lignes consécutives commencent ou finissent par le même mot ou la même série de lettres, cela peut induire le lecteur en erreur et cela est donc à éviter.

La correction automatique de ce phénomène est très complexe et en général non souhaitable. [6] C'est pourquoi l'option `homeoarchy` de ce paquet se contente de les détecter et de les afficher. Leur correction consistera en général en l'introduction d'une espace insécable dans le paragraphe:

```
\usepackage[homeoarchy]{impnattypo}
```

2. http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line
3. http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line/28361#28361
4. http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line/28358#28358
5. http://tex.stackexchange.com/questions/27780/one-letter-word-at-the-end-of-line
6. http://tex.stackexchange.com/questions/27588/repetition-of-a-word-on-two-lines

Lorsque cette option est activée, seul LuaTeX (via la commande `lualatex`) pourra effectuer le rendu du document.

Cette option n'est effective que si l'option `draft` est activée.

Les espaces insécables insérées sont colorées de deux couleurs :

- Les mots entiers sont colorés en red et cette couleur peut être ajustée par l'option `homeoarchywordcolor` ;
- Les mots partiels sont colorés en orange et cette couleur peut être ajustée par l'option `homeoarchycharcolor` ;

Une séquence de glyphes est considérée comme problématique si :

- Le nombre de mots entiers matchant est supérieur à 1. Ce paramètre peut être ajusté par l'option `homeoarchymaxwords` ;
- Le nombre de caractères matchant est supérieur à 3. Ce paramètre peut être ajusté par l'option `homeoarchymaxchars` ;

`rivers` Une lézarde est un alignement vertical d'espaces dans un paragraphe. L'option `rivers` permet de colorer les lézardes afin de les identifier. Cette option ne corrige pas les lézardes détectées :

```
\usepackage[rivers]{impnattypo}
```

Lorsque cette option est activée, seul LuaTeX (via la commande `lualatex`) pourra effectuer le rendu du document.

Cette option n'est effective que si l'option `draft` est activée.

Les espaces insécables insérées sont colorées en lime. Cette couleur peut être ajustée par l'option `riverscolor`.

2.3 Numérotation des chapitres

`frenchchapters` Concernant la numérotation des chapitres, le lexique indique : « Dans un titre, on compose en chiffres romains grandes capitales les numéros de chapitres, à l'exception de l'ordinal « premier » en toutes lettres malgré la tendance actuelle qui tend à lui substituer la forme cardinale Chapitre I. »

L'option `frenchchapters` du paquet implémente cette recommendation :

```
\usepackage[frenchchapters]{impnattypo}
```

Si vous souhaitez bénéficier de la forme ordinale « premier » sans pour autant utiliser une numérotation des chapitres en chiffres romains, il est possible de redéfinir la macro `frenchchapter`, par exemple :

```
\let\frenchchapter\arabic % numérotation en chiffres arabes
\let\frenchchapter\babylonian % numérotation en chiffres babyloniens
```

2.4 Lignes orphelines et veuves

Il est fortement recommendé de ne pas laisser de lignes orphelines dans un document. Pour cela, nous vous conseillons d'utiliser le paquet `nowidow`:

```
\usepackage[all]{nowidow}
```

Voir la documentation de ce paquet pour plus d'options.

2.5 Mode brouillon

`draft`

Le paquet impnattypo dispose d'un mode brouillon permettant de visualiser les pénalités (espaces insécables) ajoutés par les options `nosingleletter` et `lastparline`, ainsi que les informations ajoutées par les options `homeoarchy` et `rivers`. En mode brouillon, les emplacements des espaces insécables insérés sont marqués par des rectangles de couleur.

Pour activer le mode brouillon, utilisez l'option `draft`, par exemple:

```
\usepackage[draft,lastparline]{impnattypo}
```

Cette document est générée avec l'option `draft` afin d'en montrer les effets.

3 Implementation

```
\ProvidesPackage{impnattypo}
\RequirePackage{ifluatex}
\RequirePackage{kvoptions}
\SetupKeyvalOptions{
   family=impnattypo,
   prefix=int,
}
\DeclareBoolOption{draft}
\DeclareBoolOption{frenchchapters}
\DeclareBoolOption{hyphenation}
\DeclareBoolOption{nosingleletter}
\DeclareBoolOption{parindent}
\DeclareBoolOption{lastparline}
\DeclareBoolOption{homeoarchy}
\DeclareBoolOption{rivers}
\DeclareStringOption[red]{homeoarchywordcolor}
\DeclareStringOption[orange]{homeoarchycharcolor}
\DeclareStringOption[brown]{nosinglelettercolor}
\DeclareStringOption[teal]{lastparlinecolor}
\DeclareStringOption[lime]{riverscolor}
\DeclareStringOption[1]{homeoarchymaxwords}
\DeclareStringOption[3]{homeoarchymaxchars}
\ProcessKeyvalOptions*
```

No page finishes with an hyphenated word

Discourage hyphenation on two lines in a row

Number chapters

No single letter

```
\RequirePackage{xcolor}
\def\usecolor#1{\csname\string\color@#1\endcsname\space}

\ifinthyphenation
   \brokenpenalty=10000

   \doublehyphendemerits=1000000000
\fi

\ifintfrenchchapters
   \let\frenchchapter\Roman
   \renewcommand{\thechapter}{%
     \ifnum\value{chapter}=1
       premier%
     \else
       \frenchchapter{chapter}%
     \fi
   }
\fi

\ifintnosingleletter
   \ifluatex
      \RequirePackage{luatexbase,luacode}
      \begin{luacode}

      local prevent_single_letter = function (head)
        while head do
          if head.id == 37 then                                              -- glyph
            if unicode.utf8.match(unicode.utf8.char(head.char),"%a") then    -- some kind of let
               if head.prev.id == 10 and head.next.id == 10 then             -- only if we are a

                 local p = node.new("penalty")
                 p.penalty = 10000

                 \ifintdraft
                    local w = node.new("whatsit","pdf_literal")
                    w.data = "q \usecolor{\intnosingleletttercolor} 0 0 m 0 5 l 2 5 l 2 0 l b Q"

                    node.insert_after(head,head,w)
                    node.insert_after(head,w,p)
                 \else
                    node.insert_after(head,head,p)
                 \fi
               end
            end
          end
          head = head.next
        end
        return true
      end

      luatexbase.add_to_callback("pre_linebreak_filter",prevent_single_letter,"~")
```

```
      \end{luacode}
   \else
      \PackageError{The nosingleletter option only works with LuaTeX}
   \fi
\fi
```

Paragraph indentation

```
\ifintparindent
\setlength{\parindent}{1em}
\fi
```

Last line of paragraph

```
\ifintlastparline
   \ifluatex
      \RequirePackage{luatexbase,luacode}
      \begin{luacode}
      last_line_twice_parindent = function (head)
        while head do
          local _w,_h,_d = node.dimensions(head)
          if head.id == 10 and head.subtype ~= 15 and (_w < 2 * tex.parindent) then

              -- we are at a glue and have less then 2*\parindent to go
              local p = node.new("penalty")
              p.penalty = 10000

              \ifintdraft
                 local w = node.new("whatsit","pdf_literal")
                 w.data = "q \usecolor{\intlastparlinecolor} 0 0 m 0 5 l 2 5 l 2 0 l b Q"

                 node.insert_after(head,head.prev,w)
                 node.insert_after(head,w,p)
              \else
                 node.insert_after(head,head.prev,p)
              \fi
          end

          head = head.next
        end
        return true
      end

      luatexbase.add_to_callback("pre_linebreak_filter",last_line_twice_parindent,"lastparline"
      \end{luacode}
   \else
      \setlength{\parfillskip}{0pt plus\dimexpr\textwidth-2\parindent}
   \fi
\fi
```

Detect homeoarchies

```
\ifinthomeoarchy
 \ifintdraft
   \ifluatex
      \RequirePackage{luatexbase,luacode}
      \begin{luacode}
      compare_lines = function (line1,line2)
```

```
  local head1 = line1.head
  local head2 = line2.head

  local char_count = 0
  local word_count = 0

  while head1 and head2 do
     if (head1.id == 37 and head2.id == 37
            and head1.char == head2.char)          -- identical glyph
        or (head1.id == 10 and head2.id == 10) then  -- glue

        if head1.id == 37 then -- glyph
           char_count = char_count + 1
        elseif char_count > 0 and head1.id == 10 then -- glue
           word_count = word_count + 1
        end
        head1 = head1.next
        head2 = head2.next
     elseif (head1.id == 0 or head2.id == 0) then -- end of line
        break
     elseif (head1.id ~= 37 and head1.id ~= 10) then -- some other kind of node
        head1 = head1.next
     elseif (head2.id ~= 37 and head2.id ~= 10) then -- some other kind of node
        head2 = head2.next
     else -- no match, no special node
        break
     end
  end
  -- analyze last non-matching node, check for punctuation
  if ((head1 and head1.id == 37 and head1.char > 49)
       or (head2 and head2.id == 37 and head2.char > 49)) then
     -- not a word
  elseif char_count > 0 then
     word_count = word_count + 1
  end
  return char_count,word_count,head1,head2
end

compare_lines_reverse = function (line1,line2)
  local head1 = node.tail(line1.head)
  local head2 = node.tail(line2.head)

  local char_count = 0
  local word_count = 0

  while head1 and head2 do
     if (head1.id == 37 and head2.id == 37
            and head1.char == head2.char)          -- identical glyph
        or (head1.id == 10 and head2.id == 10) then  -- glue

```

```
171             if head1.id == 37 then -- glyph
172                char_count = char_count + 1
173             elseif char_count > 0 and head1.id == 10 then -- glue
174                word_count = word_count + 1
175             end
176             head1 = head1.prev
177             head2 = head2.prev
178          elseif (head1.id == 0 or head2.id == 0) then -- start of line
179             break
180          elseif (head1.id ~= 37 and head1.id ~= 10) then -- some other kind of node
181             head1 = head1.prev
182          elseif (head2.id ~= 37 and head2.id ~= 10) then -- some other kind of node
183             head2 = head2.prev
184          elseif (head1.id == 37 and head1.char < 48) then -- punctuation
185             head1 = head1.prev
186          elseif (head2.id == 37 and head2.char < 48) then -- punctuation
187             head2 = head2.prev
188          else -- no match, no special node
189             break
190          end
191       end
192       -- analyze last non-matching node, check for punctuation
193       if ((head1 and head1.id == 37 and head1.char > 49)
194            or (head2 and head2.id == 37 and head2.char > 49)) then
195          -- not a word
196       elseif char_count > 0 then
197          word_count = word_count + 1
198       end
199       return char_count,word_count,head1,head2
200    end
201
202    highlight = function (line,nend,color)
203       local n = node.new("whatsit","pdf_literal")
204
205       -- get dimensions
206       local w,h,d = node.dimensions(line.head,nend)
207       local w_pts = w/65536 -- scaled points to points
208
209       -- set data
210       n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"
211
212       -- insert node
213       n.next = line.head
214       line.head = n
215       node.slide(line.head)
216    end
217
218    highlight_reverse = function (nstart,line,color)
219       local n = node.new("whatsit","pdf_literal")
220
```

```
221
222         -- get dimensions
223         local w,h,d = node.dimensions(nstart,node.tail(line.head))
224         local w_pts = w/65536 -- scaled points to points
225
226         -- set data
227         n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"
228
229         -- insert node
230         node.insert_after(line.head,nstart,n)
231       end
232
233       homeoarchy = function (head)
234         local cur_line = head
235         local prev_line -- initiate prev_line
236
237         local max_char = tonumber(\inthomeoarchymaxchars)
238         local max_word = tonumber(\inthomeoarchymaxwords)
239
240         while head do
241           if head.id == 0 then -- new line
242             prev_line = cur_line
243             cur_line = head
244             if prev_line.id == 0 then
245                -- homeoarchy
246                char_count,word_count,prev_head,cur_head = compare_lines(prev_line,cur_line)
247                if char_count >= max_char or word_count >= max_word then
248                   local color
249                   if word_count >= max_word then
250                      color = "q \usecolor{\inthomeoarchywordcolor}"
251                   else
252                      color = "q \usecolor{\inthomeoarchycharcolor}"
253                   end
254
255                   -- highlight both lines
256                   highlight(prev_line,prev_head,color)
257                   highlight(cur_line,cur_head,color)
258                end
259             end
260           end
261           head = head.next
262         end
263         return true
264       end
265
266       luatexbase.add_to_callback("post_linebreak_filter",homeoarchy,"homeoarchy")
267
268       homoioteleuton = function (head)
269         local cur_line = head
270         local prev_line -- initiate prev_line
```

```
271
272         local max_char = tonumber(\inthomeoarchymaxchars)
273         local max_word = tonumber(\inthomeoarchymaxwords)
274
275         local linecounter = 0
276
277         while head do
278           if head.id == 0 then -- new line
279             linecounter = linecounter + 1
280             if linecounter > 1 then
281                prev_line = cur_line
282                cur_line = head
283                if prev_line.id == 0 then
284                   -- homoioteleuton
285                   char_count,word_count,prev_head,cur_head = compare_lines_reverse(prev_line,cu
286                   if char_count >= max_char or word_count >= max_word then
287                      local color
288                      if word_count >= max_word then
289                         color = "q \usecolor{\inthomeoarchywordcolor}"
290                      else
291                         color = "q \usecolor{\inthomeoarchycharcolor}"
292                      end
293
294                      -- highlight both lines
295                      highlight_reverse(prev_head,prev_line,color)
296                      highlight_reverse(cur_head,cur_line,color)
297                   end
298                end
299             end
300           end
301           head = head.next
302         end
303
304         return true
305       end
306
307       luatexbase.add_to_callback("post_linebreak_filter",homoioteleuton,"homoioteleuton")
308       \end{luacode}
309     \else
310       \PackageError{The homeoarchy option only works with LuaTeX}
311     \fi
312   \fi
313 \fi
```

Detect rivers

```
314 \ifintrivers
315   \ifintdraft
316     \ifluatex
317       \RequirePackage{luatexbase,luacode}
318       \begin{luacode}
319 river_analyze_line = function(line,dim1,dim2)
```

```
   local head = line.head

   while head do
      if head.id == 10 then  -- glue node
         local w1,h1,d1 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.hea
         local w2,h2,d2 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.hea
         --print("dim1:"..dim1.."; dim2:"..dim2.."; w1:"..w1.."; w2:"..w2)
         if w1 > dim2 then  -- out of range
            return false,head
         elseif w1 < dim2 and w2 > dim1 then -- found
            return true,head
         end
      end
      head = head.next
   end

   return false,head
end

rivers = function (head)
   local prev_prev_line
   local prev_line
   local cur_line = head
   local cur_node
   local char_count

   local linecounter = 0

   while head do
      if head.id == 0 then -- new line
         linecounter = linecounter + 1
         prev_prev_line = prev_line
         prev_line = cur_line
         cur_line = head
         if linecounter > 2 then
            cur_node = cur_line.head
            char_count = 0

            while cur_node do
               if cur_node.id == 37 then  -- glyph
                  char_count = char_count + 1
               elseif cur_node.id == 10 and char_count > 0 and cur_node.next then  -- glue node
                  local w1,h1,d1 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order
                  local w2,h2,d2 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order
                  found_pp,head_pp = river_analyze_line(prev_prev_line,w1,w2)
                  found_p,head_p = river_analyze_line(prev_line,w1,w2)

                  if found_pp and found_p then
                     local n_pp = node.new("whatsit","pdf_literal")
                     n_pp.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 l 5 5 l 5 0 l b Q"
```

```
                    node.insert_after(prev_prev_line,head_pp.prev,n_pp)

                    local n_p = node.new("whatsit","pdf_literal")
                    n_p.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 l 5 5 l 5 0 l b Q"
                    node.insert_after(prev_line,head_p.prev,n_p)

                    local n_c = node.new("whatsit","pdf_literal")
                    n_c.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 l 5 5 l 5 0 l b Q"
                    node.insert_after(cur_line,cur_node.prev,n_c)
                end
            end
            cur_node = cur_node.next
        end
      end
    end
    head = head.next
  end

  return true

end


luatexbase.add_to_callback("post_linebreak_filter",rivers,"rivers")
    \end{luacode}
  \else
    \PackageError{The homeoarchy option only works with LuaTeX}
  \fi
\fi
\fi
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.