BIJLAGE R

The Development of National LaTeX styles

Johannes Braams, Victor Eijkhout, and Nico Poppelier

revision 4 August 1989

Abstract

In its autumn 1988 meeting, the Dutch TeX users group (NTG) established a working group (number 13) that was to concentrate on the problems involved in the use of TeX for Dutch texts. Since then the working group, which includes the authors, has created a number of style options for LaTeX that remedy some common problems with the non-English use of LaTeX, and is on its way developing document styles that are compatible with the standard styles, but have a layout that is more palatable for Dutch users. In this article we treat implementation aspects of the styles and style options, and we discuss some matters of layout.

1 The need for national TeX

On several occasions it is stressed in both the TeX book [1] and the LaTeX book [2] that non-English users of TeX may have to take steps in order to adapt TeX to their native language. For several languages such steps have indeed been taken, for instance for the German language [3]. It was only natural that the Dutch TeX users group (NTG) would also initiate an effort in this direction. Thus the active life of working group 13 began somewhere about the beginning of 1989.

As the use of LaTeX is quite wide-spread in the Netherlands, and because most matters of national standardization can be conveniently handled in the context of document styles – and also because the Dutch language does not have the problems of national characters that are prominent for TeX users to the North, South and East of this country – it was decided to focus mainly on the development of national styles and style options for LaTeX. This article treats some of the problems encountered and the way they were solved.

2 The 'chapter' problem

One of the first problems non-English users of LaTeX run into, is that of English terms ('Abstract', 'Contents') contained in the document styles. The resourceful user, or the TeXnician consulted, will probably take out a text editor and hunt through the style file for the offending string, replacing it by its equivalent in his/her native language. This process will most likely result in new styles, called (for Dutch) 'artikel', 'rapport' and so on.

At this point it may occur to the conscientious user that Leslie Lamport must have foreseen this situation, and probably have made provisions for it, so why not see if there is a suggested approach for this. A cursory perusal of the table of contents of the LaTeX book will then lead our user to section 5.1.4 'Customizing the document style'. There Lamport takes half a page to explain how the word 'Chapter' is really just the value of a control sequence `\@chapapp`. This implies that changing this text to 'Hoofdstuk' does not require editing of document styles at all, it merely needs a one-line option file. Charming, one would say. However, immediately after that there is a strange sentence 'You may also want to redefine the `\appendix` command, replacing `Appendix [...]`'. Curiouser and curiouser! Can't I just redefine some control sequence that yields the word 'Appendix'? Well, as it turns out, 'Chapter' is the *only* text that has been parametrized, the rest is hard-wired into the document styles.

And such is the situation in which working group 13 found itself: we could think of at least three ways of solving the 'Chapter' problem.

- We could make exact copies of the standard styles and all point size options ('art10.sty' and such), replacing all English text by Dutch text. The disadvantages of this are that (1) supposing a certain installation of TeX will be used for three languages, then every style file has to be present three times, and (2) if Lamport then finds a bug or decides to issue upgrades of the standard styles this would require a multitude of changes.
- We might also rewrite the standard styles, parametrizing them, and add option files for the different languages that only contain parameter settings. This solution is only subject to the second objection above.
- In fact there exists an even beter solution (due to Piet van Oostrum): if an option file would be able to find out what style is being used, it can replace and parametrize just those commands that contain text, and afterwards set the parameter values to some appropriate language. This approach is probably the most economical one: the original styles are still used, and, if the option file contains parameter settings for a number of languages, this single option file suffices.

The option file 'dutch', created by one of the authors (JB), implements the third possibility above. It uses for parameter names those suggested by Hubert Partl [3] in the 'german' style. Note however, that 'german' – an implementation of the second possibility above – does not do the actual redefinitions but merely sets the parameters. Thus it will only function correctly with edited standard styles.

As an example of parametrization of commands consider the following definition, taken from `article.sty`[1]

```
\def\abstract{\if@twocolumn
 \section*{Abstract}
\else \small
 \begin{center}
 {\bf Abstract\vspace{-.5em}}
 \end{center}
 \quotation
\fi}
```

This definition is overridden in `dutch.sty` by

```
\def\abstract{\if@twocolumn
 \section*{\abstractname}
\else \small
 \begin{center}
 {\bf \abstractname\vspace{-0.5em}}
 \end{center}
 \quotation
\fi}
```

to which is added a command initializing the language-dependent parameters:

```
\def\captionsdutch{
\def\abstractname{Samenvatting}
.
.
}
```

Some comments about this approach are in order. First of all, the option file `dutch.sty` really consists of two disparate parts, the redefinitions and the initializations. The redefinitions are only to make up for

[1]The macros presented here have been simplified to convey the essence of what we are telling. They are not usable in this form.

what we see as a deficiency in the distribution style files. The parameter initializations are then the truly language-dependent part.

Secondly, the option ‘dutch’ obviously works with the standard document styles, but will collide with some other styles, for instance with new styles to be developed – as we found out to our chagrin. The reason for this is that the same mechanism that repairs the ‘article’ style, will attempt to repair any style that looks like it. Thus, if a Dutch ‘artikel’ style has its own ideas about how an abstract should look, it must have a way of protecting itself against ‘dutch’s zeal. Such a possibility exists, and it has been incorporated in the new Dutch styles that will be discussed below. Other non-standard styles may need to be edited before they can be used with the ‘dutch’ option, however.

3 Making the layout less ‘loud’

To Dutch – and probably some other – eyes, the LaTeX styles are a bit ‘loud’. Therefore two style options have been developed by one of the authors (NP), that work together with the standard styles, and that make the general layout a bit more compact. A first option is `a4.sty` which sets various parameters in order to accomodate European standard A4 paper. This option started out as a bare union of the ‘a4’ style option of John Pavel and the ‘a4wide’ style option of Jean-François Lamy, but the current version has undergone fine tuning. The second option is called ‘sober’: it reduces the sizes of fonts in section headings, and it eliminates the white spaces surrounding section headings and items in list structures.

4 Compatible replacement styles for Dutch

Without asserting that there exists such a thing as ‘a typically Dutch layout’, we can still state that certain aspects of the LaTeX styles are less desirable for Dutch documents. Therefore the working group at its first meeting already declared it a goal to develop styles with a Dutch look. With the guidance of a graphical designer [4], and using some books on the subject [5, 6, 7] – including one by a Dutch typographer – two styles have since then reached completion, implemented by one of the authors (VE). One style is compatible with ‘article’, and one with ‘report’. By compatibility we mean here that the new style implements the same commands as one of the original styles. Both styles are based on the same graphical design. It is our intention to have further compatible styles available in the near future based on different layouts.

4.1 The flexibility of LaTeX

Probably the easiest way to develop a new document style is to start out with an already existing one, and to modify it gradually. In this process one discovers that LaTeX has a lot of possibilities for modication built into it. For instance, whether or not the fifth parameter of `\@startsection` – the generic command used in document styles to define section headings – is positive controls the placement of the section heading above or embedded in the text. The absolute value of this parameter is then either

- the vertical distance between section heading and the first line of the text when it is positive, or
- when negative it is the horizontal distance between the run-in heading and the first word of the text.

On the other hand a number of design decisions have been hard-wired into the sectioning commands, and cannot be easily changed, for instance, by means of the parameters of `\@startsection`. Consider as an example the distance between the section number and the heading. This turns out to be exactly ‘1 em’ in the font of the section heading, as can be seen from the following definition – again, this is a rather simplified form of the definition actually appearing in `latex.tex`.

```
\def\@sect#1#2#3#4#5#6[#7]#8{
    \refstepcounter{#1}
        % #1 is 'subsection' for example
```

```
\edef\@svsec{\csname the#1\endcsname
             \hskip 1em }
\begingroup#6\relax  % #6 is the style
  \@hangfrom{\hskip #3\relax\@svsec}%
            {\interlinepenalty=\@M
             #8\par}
    % #8 is the heading text
\endgroup \@xsect{#5}}
```

which one can find, buried somewhere deep down in `latex.tex`.

Clearly, LaTeX makes it easy for the user to determine whether the section number be set using digits or roman numerals or letters: the only action required is redefinition of `\thesection` and so on. Also it is easy for the document-style designer to specify the style of the section heading: this is determined by the sixth parameter of `\@startsection`. It is not easy to change that ‘1 em’, which one might just want to do occasionally.

4.2 Going Dutch

The styles ‘artikel’ and ‘rapport’ embody a number of changes with respect to ‘article’ and ‘report’. Some of these are fairly trivial, such as the fact that we switch on ‘french spacing’, and some are more complicated. In the rest of this section we will treat the most significant change, the notion of a ‘unit indent’, both from a typographical and implementer’s point of view.

One of the principles of document design is[2] that the eye is able to pick up regularities in a page layout, and that their presence is considered positive, but that having too many is confusing. For instance, the designer we consulted insisted that the white space separating a section heading and the following text should bear some simple relation to the baselineskip, and should not have any stretch.

One point that all our sources seemed to agree on was that the number of ‘implied left margins’ in a document should be as low as possible. By an implied left margin we mean here a non-zero distance from the actual left margin that is taken by more than one item of the document. Examples of implied left margins are

- first of all the paragraph indentation, furthermore
- the left margins of items in an ‘itemize’ or ‘enumerate’ list construct, and
- the left (or right) sides of the numbers and labels in such list constructs, but also
- the left side of the text of a section heading.

In the standard styles of LaTeX all of these four distances are independent and different from each other. In the style we have developed it was decided to strengthen the visual coherence of the layout by taking the same value for each of them whenever as possible.

Implementing this idea meant adopting a new dimension `\unitindent` which first of all unifies the `\parindent` and the `\leftmargini`, the indentation of non-embedded lists.

```
\newdimen\unitindent
{\setbox0=\hbox{\normalsize\rm 2.2.2
                \hskip.5em}
 \global\unitindent=\wd0}
\parindent=\unitindent
\leftmargini=\unitindent
```

Admittedly this will give a rather large indentation, but this does not seem to be uncommon in contemporary typographical design. In fact, while the LaTeX book takes the classical quad – the Dutch term translates to ‘a square of white’ – for the paragraph indent, the TeX book shows a large indentation which is equal to that for lists on the outer level.

[2] Or rather: seems to be. We have not encountered explicit statements to this effect, but implicitly it seems to be there.

From the computation of the size of the unit indent the reader may have gathered already that it will also be put to another use: we want the text of section headings – of all numbered sections – to appear at a distance of `\unitindent` from the left margin. As was indicated above, this requires some modification of the `\@sect` macro. We therefore include in the style file the following redefinition:

```
\def\@sect#1#2#3#4#5#6[#7]#8{
    \refstepcounter{#1}
        % #1 is 'section' for example
    \edef\@svsec{\hbox to \unitindent
          {\csname the#1\endcsname \hfil}}
    \begingroup #6\relax
        % #6 is the style
      \@hangfrom{\hskip #3\relax\@svsec}%
                  {\interlinepenalty=\@M
                   \hyphenpenalty=\@M
                   \exhyphenpenalty=\@M
                   \rightskip=0cm plus 13cm
                   #8\par}
        % #8 is the heading text
    \endgroup \@xsect{#5}}
```

where the macro `\@svsec` for the heading now gives a horizontal box of a predetermined width. Note that we have also ensured that the heading is set ragged right. Hyphenation in a heading – even of words with an explicit hyphen – is something so hideous, that we suspect this to be an oversight of Leslie Lamport.

A further unification of implied left margins can be achieved if we look more carefully at embedded lists. In the standard styles an embedded list construct defines two left margins: the left margin of the items, and the left side of the labels. This second margin can be eliminated by making it equal to the text margin of the surrounding list. The placement of such labels is governed by the value of `\labelwidth` and by the macro `\makelabel` which is passed by `itemize` and `enumerate` to `\list`. We therefore specify that the label will take the full width of the indentation, for instance

```
\def\@listii{\leftmargin=\leftmarginii
 \labelwidth=\leftmarginii}
```

and redefine `\itemize` (in the document style file)

```
\def\itemize{\advance\@itemdepth \@ne
   \edef\@itemitem{\labelitem
           \romannumeral\the\@itemdepth}%
   \list{\csname\@itemitem\endcsname}%
        {\def\makelabel##1{##1\hfil}}}
```

so that it will make labels that are flush left. The result of these actions can be seen in figures 1 and 2.

4.3 The international feel

It has occurred to us that LaTeX users outside the Netherlands may also like the new styles, and may also want to use them. We decided therefore to make the styles in a sense truly compatible to 'article' and 'report': when used on their own the styles will produce English captions, and using them in combination with 'dutch' or 'german' will alter these. However, the reader will understand after the above discussion that we achieve this by somewhat more sophisticated means than are employed in the standard styles.

None of the commands in the 'artikel' style, for instance, contain actual texts. Instead they contain such commands as `\abstractname`. These commands are initialized at the end of the file to give English texts. Any language option based on the same parameter names, such as 'dutch' or 'german', can override these settings for use in other languages. Naturally, the 'artikel' style takes precautions to prevent 'dutch' from mistaking it for 'article'.

On a more philosophical note, we may add that we feel that this situation is how it should have been from the beginning. In fact, Leslie Lamport himself has shown the way to international styles – see the passage quoted above. For some reason however, he stopped short and did not implement this idea.

5 Conclusion

It has turned out to be possible to adapt LaTeX for use with the Dutch language, in such a way that hardly any action on the part of the user is required. With some style options English terms can be replaced, and the typically American layout can be made more acceptable to Dutch eyes. The development of completely new document styles for Dutch, however, leaves the implementer with the feeling that he has had to unearth sections of LaTeX that were never meant to be rewritten.

References

[1] Donald Knuth, *The TeXBook*, Addison-Wesley, 1986.

[2] Leslie Lamport, LaTeX*, A Document Preparation System*, Addison-Wesley, 1986.

[3] Hubert Partl, *German TeX*, Tugboat, vol 9 (1988), no 1, pp. 70–72.

[4] Inge Eijkhout, Conversations and correspondence, 1989.

[5] John Miles, *Ontwerpen voor Desktop Publishing, Alles over layout en typografie op de personal computer*, Uitgeverij Mingus, Baarn 1988, Dutch translation of *Design for Desktop Publishing: a guide to layout and typography on the personal computer*, Fraser, London 1987.

[6] Stanley Morison, *Grondbeginselen der Typografie*, Uitgeverij de Buitenkant, Amsterdam 1987, Dutch translation of *First Principles of Typography*.

[7] Karel Treebus, *Tekstwijzer, een gids voor het grafisch verwerken van tekst*, SDU uitgeverij, The Hague 1988.

BIJLAGE S

Unusual paragraph shapes

Victor Eijkhout

Introduction

Although the TeX book [1] states that TeX's paragraph mechanism 'can be harnessed to a surprising variety of tasks', the strangest paragraph shapes that I have implement use no feature of the line-breaking algorithm. Instead, I have found that the control sequences `\everypar` and `\lastbox` are extremely powerful tools. I give three examples of this.

How many TeXers does it take to typeset a lightbulb? [2]

A while back I found somewhere in Netland a largish collection of jokes – several hundreds of them – consisting of a question of the above form, and an answer to it. It seemed to me that this document would make a nice gift, if properly typeset and bound. The following layout was suggested to me [3]:

How many evolutionists does it take to screw in a light bulb?

Only one, but it takes eight million years.

How many folk singers does it take to screw in a light bulb?

Two. One to change the bulb, and one to write a song about how good the old light bulb was.

As all jokes[1] had a question mark concluding the first line, and the file had blank lines seperating the jokes, the following macro should do the job when placed in front of each joke:

```
\def\lbj#1? #2\par{{\it
    \hangfrom{\vrule width 1in height 1ex
              depth 0in \kern .5em}#1\par}
    #2\par}
```

Only I didn't feel like putting some command in front of a couple of hundreds of jokes. Then I remembered about `\everypar`, the token list that is inserted in front of every paragraph. If I would set

```
\everypar={\lbj}
```

the jokes would automatically be scooped up as the arguments of `\lbj`.

Well, *almost.* The snag is that `\lbj` itself produces two paragraphs, in front of each of which `\everypar` is inserted. With infinite recursion as a result. Repair may take the following form:

```
\everypar={\bgroup\everypar={}\lbj}
\def\lbj#1? #2\par{ ...  %% the same
    ...                  %% as above
    #2\par\egroup\bigskip}
```

Now `\everypar` opens a group inside which it is zero, and `\lbj` inserts the closing brace of the group, so after the joke `\everypar` has its old value again.

[1] I guess the answer to the above question is: they are still waiting for the `dvi2lightbulb` program

A Maserati, a Mack Truck! [4]

In the previous example, the `\everypar` takes the whole paragraph as its argument. This is not a wise approach, however, as it places memory demands on TeX in case of long paragraphs, and it means that the text is scanned twice. One may then define

```
\everypar{\catchthepar\everypar={}}
\def\par{\endgraf\egroup}
```

with for instance

```
\def\catchthepar{\vbox\bgroup}
```

I took this approach when I implemented 'vario setting' of text, a paragraph layout where lines are right adjusted, except when they have to stretch beyond some point. In that case they remain 'ragged right'. This way of setting text has in recent years become very fashionable in advertisements, especially for automobiles. Provided the paragraphs are suitably long, say five lines or more, and the right proportion of lines is left unadjusted, the visual effect of a block with an occasional gap in the right margin can be, well, interesting. Some typographers, however, loathe this concoction [3].

When I first tried to implement this, I fiddled around with the usual parameters, but that led to nothing. Also I didn't want to take the approach from [5], as this would preclude hyphenated words. The only other way that I could think of, was to let TeX set the paragraph right adjusted, and afterwards to unset the glue in some lines. Getting the lines from the paragraph would be possible, I thought, using `\lastbox`.

As it turned out there were two difficulties. The first one was that lines that hang on the main vertical list can't be picked with `\lastbox`. I solved this by a variation of the above construct:

```
\everypar={\vbox\bgroup
    \everypar={}
    \def\par{\endgraf\pickthelines
             \egroup}}
```

Implementing `\pickthelines` brought to light the second difficulty: successive calls to `\lastbox` yield void boxes after the first call. At least, this was how it looked to me at first.

After a while I realised that the real reason was that a paragraph consists of an alternation of a box, a penalty, and a glue item, and so on. So I arrived at

```
\def\pickthelines{
    \setbox\investigation=\lastbox
    \ifvoid\investigation \else
    \unskip \savepenalty=\lastpenalty
    \unpenalty
       {\pickthelines}
    \investigatethebox
    \penalty\savepenalty \fi}
\def\investigatethebox
   {\setbox\tester=\hbox
           {\unhcopy\investigation}
    \ifdim\wd\tester<.94\hsize
          \box\tester
    \else \box\investigation \fi}
```

The recursion in `\pickthelines` uses TEX's save stack, so this macro is not suited for very long paragraphs. But that's not what it is intended for.

Note that TEX automatically reinserts the correct baselineskips, but that I have to do the appropriate penalties myself. The value beyond which a line should 'snap back' has to be determined experimentally; it highly depends on the `\hsize`, and the average word length of the text. Also, in order to prevent multi-line gaps, a test should be implemented if the previous line was set at natural width. Further fine tuning may involve the `\spaceskip` and the `\tolerance`.

Romeo, wherefore art thou indented? [6]

Browsing through Shakespeare's complete works, I lighted upon a phenomenon that I thought would be pretty hard for TEX. Rather than explaining it at length, I'll just show an example.

Jul. Romeo!
Rom. My dear!
Jul. At what o'clock to-morrow

shall I send to thee?
Rom. At the hour of nine.
Jul. I will not fail; 'tis twenty years till then.

The input for this example is:

```
\open
\Jul  Romeo!
\Rom[ My dear!
\Jul[ At what o'clock to-morrow
      shall I send to thee?
\Rom[ At the hour of nine.
\Jul  I will not fail; %
'tis twenty years till then.
\close
```

The reason for the strange indentation lies in a form principle of Shakespeare's plays: all lines are in iambic meter, which implies that the number of syllables is somewhat predetermined. As this number is nowhere near reached after an exclamation like 'Romeo!', the addressed party will have to finish the line for Juliet. He falls far short of this ideal, by the way.

For this paragraph shape I didn't use `\everypar`, but `\lastbox` would certainly be needed, namely to measure the last line of the previous player. In order to be able to use `\lastbox` all the text would have to be included in `\vboxes`. Each player should then do the following:

- measure the last line of the previous player,
- close the text of that player and hang it to the page, and
- then indent by the measured amount if a square bracket follows.

With some precautions for the first and last line of a scene, the basic commands look like:

```
\def\player#1 {\expandafter
    \def\csname#1\endcsname
       {\global\def\name{#1.}\playerline}}
\player Rom \player Jul
\newif\ifplaying % action started yet?
\def\open{\obeylines \playingfalse}
\def\close{\closepreviousplayer}
\def\playerline{\ifplaying
    \closepreviousplayer
    \else \playingtrue \fi
```

```
    \futurelet\next\maybeindentedline}
```

The job of 'closing the previous player' consists of taking the last box, measuring it, and hanging it back on the list. I leave the details to the reader.

```
\def\closepreviousplayer
   {\getpromptlength\egroup\unvbox\lines}
```

After the previous player has been closed, a new one can be opened, i.e., the box `\line` can be opened, with an indentation if necessary.

```
\def\maybeindentedline{\if\next[
    \expandafter\eatbracket
    \else \global\promptlength=0cm
          \nonindentedline \fi}
\def\eatbracket[{\nonindentedline}
\def\nonindentedline{\setbox\lines=
    \vbox\bgroup\leavevmode\strut
        {\setbox\tempbox=\hbox{\it\name\enspace}%
         \ifdim\promptlength>\wd\tempbox
               \hbox to \promptlength
                    {\it\name\hfil}%
         \else \box\tempbox \fi}}
```

For the complete layout used in the edition of Shakespeare's plays I have these macros will have to be augmented by provisions for narrow columns: a too long line is split with the remainder set flush right.

Conclusion

Aberrant paragraph shapes can be automated in TEX to such an extent that no commands need to be inserted in the actual text. The token list `\everypar` is a handy tool for this, and for applications involving the length of lines `\lastbox` is a powerful instrument.

References

[1] Donald Knuth, The TEX book, 1986.

[2] Anonymous, The canonical collection of light bulb jokes.

[3] Inge Eijkhout, conversations, 1988/9.

[4] The Tubes, What do you want from life, 1975.

[5] Anne Brüggeman-Klein, Tugboat, 1988.

[6] William Shakespeare, Romeo and Juliet, 1594.

BIJLAGE T

Ingekomen brief

TEXnique

Bureau voor TEXpertise
en cursusontwikkeling

TEXnique • Washingtondreef 153 • 3564 KD Utrecht

secretaris NTG
Gerard van Nes
ENR Technisch-wetenschappelijk Rekencentrum
Westerduinweg 3
1755 ZG Petten

Uw brief van	Uw kenmerk	Ons Kenmerk	Datum
–	–	np-891116	16 november 1989

Onderwerp: LATEX-cursussen

Beste Gerard,

Als direct gevolg van mijn activiteiten op TEX-gebied in de afgelopen twee jaar, onder andere de cursus documentstijlontwikkeling die ik op 29 juni van dit jaar gegeven heb, ben ik op 1 september 1989 begonnen met een eigen bureau, TEXnique en daarvan wilde ik ook de NTG op de hoogte brengen. De activiteiten van TEXnique zullen de volgende drie zaken omvatten

- ontwikkeling van LATEX-cursussen,
- adviezen over TEX en LATEX en
- ontwikkeling van documentstijlen.

De bedoeling is dat ik vanaf omstreeks april 1990 de volgende cursussen kan verzorgen

- LATEX voor gevorderden (LATEX.200) en
- documentstijlontwikkeling (LATEX.400).

Over de inhoud van deze cursussen is overlegd met de cursus-coördinator van *TUG*.

Hoogachtend,

Nico Poppelier

KvK Utrecht
92958

telefoon
030-625105