

BIJLAGE CC

An indentation scheme

Victor Eijkhout

University of Illinois at Urbana-Champaign

305 Talbot Lab

104 S. Wright Street

Urbana, Illinois 61801, USA

u641001@HNYKUN11

Indentation is one of the simpler things in \TeX : if you leave one input line open you get a new paragraph, and it is indented unless you say `\noindent`. And if you get tired of writing `\noindent` all of the time, you declare

```
\parindent=0pt
```

at the start of your document. Easy.

More sophisticated approaches to indentation are possible, however. In this article I will sketch a quite general approach that can easily be incorporated in existing macro packages. For a better appreciation of what goes on, I will start with a tutorial section on what happens when \TeX starts a paragraph.

1 Tutorial: paragraph start

When \TeX is not busy typesetting mathematics, it is processing in *horizontal mode*, or *vertical mode*. In horizontal mode it is putting objects – usually characters – next to each other, in vertical mode it is putting objects – usually lines of text – on top of each other.

To see that there is a difference, run the following pieces of code through \TeX :

```
\hbox{a}
\hbox{b}
\bye
```

and

```
a
\hbox{b}
\hbox{c}
\bye
```

You notice that the same objects are treated in two different ways. The reason for this is that \TeX starts each job in vertical mode, that is, stacking material. In the second piece of input \TeX saw the character ‘a’ before it saw the boxes. A character is for \TeX the sign to switch to horizontal mode, that is, lining up material, and start building a paragraph.

Commands that can make \TeX switch to horizontal mode are called ‘horizontal commands’. As appeared from

the above two examples characters are horizontal commands, but boxes are not. Let us now look at the two most obvious horizontal commands: `\indent` and `\noindent`.

1.1 `\indent` and `\noindent`

`\indent` is the command to start a paragraph with indentation. \TeX realizes the indentation by inserting a box of width `\parindent`. If you say `\indent` somewhere in the middle of a paragraph you get some white space there, caused by the empty box.

`\noindent` is the command to start a paragraph without indentation. After this command \TeX merely switches to horizontal mode; no indentation box is inserted. If you give this command somewhere in the middle of a paragraph it has no effect at all.

If \TeX sees a horizontal command that is not `\indent` or `\noindent`, for instance a character, it acts as if the command was preceded by `\indent`. This is why paragraphs usually start with an indentation.

As an illustration here is a small variation on the above two examples:

```
\noindent
\hbox{a}
\hbox{b}
```

```
\indent
\hbox{a}
\hbox{b}
\bye
```

1.2 `\everypar`

\TeX performs another action when it starts a paragraph: it inserts whatever is currently the contents of the token list `\everypar`. Usually you don’t notice this, because the token list is empty in plain \TeX (the \TeX book [3] gives only a simple example, and the exhortation ‘if you let your imagination run you will think of better applications’). \LaTeX [5], however, makes regular use of `\everypar`. Some mega-trickery with `\everypar` can be found in [2].

- Just to show how this works, I put in front of this paragraph the statement

```
\everypar={\bullet\quad}
```

That is, I told T_EX that `\bullet\quad` should be inserted in front of a paragraph.

- There's nothing specified for this paragraph; I get the bullet for free, as `\everypar` does exactly what its name promises: it is inserted in front of *every* paragraph.

At the end of the previous paragraph I specified

```
\everypar={}
```

so nothing is inserted from this paragraph onwards.

1.3 Removing indentation

Every T_EX user knows that indentation can be prevented globally by setting `\parindent` to zero. However, this is rather crude, and if you use the plain T_EX macros you may notice several rather unpleasant side effects of this action, for instance when you use the macros `\item` and `\footnote`.

It is possible to use `\everypar` to prevent indentation, or more correctly: to remove indentation. This can be achieved by

```
\everypar={{\setbox0=\lastbox}}
```

This needs some explanation.

If the last item that was processed by T_EX is a box, then that box is accessible by the command `\lastbox`. If the last item was not a box then `\lastbox` is an empty box, but no error ensues. As the `\everypar` list is inserted after any indentation box, the `\lastbox` command will get hold of the indentation box if there is one. By assigning the last box to another box register – here `\box0` – it is removed from where it was previously.

Finally, the statement

```
\setbox0=\lastbox\
```

is enclosed in braces. T_EX's grouping mechanism restores values when the group ends that were current when the group began. In this case it has the effect of totally removing the indentation box: first it is taken and assigned to `\box0`, then the value of `\box0` is restored to whatever it was before the group began.

1.4 Other actions at the start of a paragraph

In the above discussion I have omitted one action that takes place at the start of a paragraph: T_EX inserts (vertical) `\parskip` glue above the paragraph. As this has no relevance for the subject of indentation I will not go into it any further. However, in a subsequent article I will give more information about `\parskip`.

2 To indent or not to indent

In classical book typography [4] every paragraph is indented, with the exception of the first paragraph of a chapter. Nowadays a design where no paragraph indents is quite common. There are two mixtures between always indenting and never indenting: occasionally indenting, and occasionally not indenting. Thus it seems possible to characterize indentation strategies by two yes/no parameters: one that decides whether paragraphs should indent in principle, and another parameter that can overrule those decisions. Let us now see how this can be implemented in T_EX.

2.1 Implementation

Above I have already indicated that changes to `\parindent` should be avoided. Let us then assume that `\parindent` is greater than zero, even if we will never indent a paragraph (see [1] for other uses for the `\parindent` quantity). We must then realize unindented paragraphs by removing their indentation as explained above.

First we need a macro for removing the indentation:

```
\def\removeindentation
  {\setbox0=\lastbox}
```

Then we need the switches that control indentation:

```
\newif\ifNeedIndent %as a rule
\newif\ifneedindent %special cases
```

Now for the definition of `\everypar`. This is a bit tricky.

Let us first collect some bits and pieces. The main question is to decide when `\removeindent` should be called. This is for instance the case if `\NeedIndentfalse`, and that parameter is not overruled by `\needindenttrue`.

```
\ifNeedIndent
  \ifneedindent
  \else \removeindentation
\fi \fi
```

Indentation should also be removed in case `\NeedIndenttrue`, but when that parameter is overruled by `\needindentfalse`.

```
\ifNeedIndent
\else \ifneedindent
  \else \removeindentation
\fi \fi
```

Next we should make sure that `\ifneedindent` is used only for exceptional cases: if the user or a macro sets this parameter to a different value from `\ifNeedIndent`, then that should be obeyed exactly

once.

```
\ifNeedIndent
  \ifneedindent
  \else \needindenttrue \fi
\else \ifneedindent \needindentfalse
\fi \fi
```

This is then the full definition of `\everypar`:

```
\everypar={\controlledindentation}
\def\controlledindentation
  {\ifNeedIndent
    \ifneedindent
    \else \removeindentation
         \needindenttrue
    \fi
  \else \ifneedindent
         \needindentfalse
    \else \removeindentation
  \fi \fi}
```

Another implementation would be possible:

```
\def\controlledindentation
  {\ifneedindent
  \else \removeindentation \fi
  \let\ifneedindent=\ifNeedIndent}
```

This saves one conditional, but for most paragraphs it involves an unnecessary `\let` command.

2.2 Usage

My aim in developing this indentation scheme was to hide all commands pertaining to indentation in macros. The user should have to specify only once whether paragraphs should indent as a rule:

```
NeedIndenttrue
```

and then macros should declare the exceptions:

```
\def\section#1{...
  \needindentfalse
  ...}
```

2.3 But couldn't you simply ... ?

Maybe people who read this have written macros themselves that end like

```
\def\section#1{...
  ...
  \noindent}
```

or

```
\def\section#1\par{...
  ...
  \noindent}
```

This works reasonably well, but it is not completely safe. In the first case there shouldn't be an empty line after a

```
\section{...}
```

call, and in the second case there can only be one empty line after

```
\section ...
```

The reason for this is that *every* empty line generates a `\par` command, which annuls the effect of the `\noindent`. Hence the more drastic approach.

An argument the other way around can also be found, by the way. As Ron Whitney pointed out to me, the following piece of code causes trouble:

```
\section{Title}%
  {\smallcaps The first} words are ...
```

Any changes made by `\everypar` are now effected *inside a group*. In this case one remedy is to insert a `\leavevmode` command, or to define

```
\def\smallcapswords#1{\leavevmode
  {\smallcaps #1}}
```

which can be used at any place.

Another remedy would be to let all assignments controlling indentation be global. However, there are some subtle objections to this.

3 About macro packages and users

Above I remarked that plain \TeX does not use `\everypar`, and that \LaTeX redefines it a lot. This means that in plain \TeX the user is free to take every value of `\everypar` that he or she likes; in \LaTeX every attempt of the user to use `\everypar` is immediately thwarted.

One might ask how the use of `\everypar` that I have sketched compares to this. Can the user be allowed to access `\everypar`, even if the macro package needs it all the time?

In my own 'Lollipop' format I have taken the following way out. The user or the style designer is allowed to fill in `\everypar`, as long the statement

```
\the\everyeverypar
```

is included. Here `\everyeverypar` is the token list with the constant actions such as indentation control that should be performed always.

A format designer who wishes to hide even this from the user or the style designer, could use the following piece of code

```
\newtoks\temppar
\def\everyparagraph
  {\afterassignment\evpar
  \temppar}
```

```
\def\xevpar
  {\edef\act{\everypar=
    {\the\temppar
     \the\everyeverypar
    }}%
  \act}
```

so that it becomes possible to write

```
\everyparagraph={\DoSomething
  \everyparagraph={}}
```

while the `\everypar` will still contain all of the constant actions.

Short explanation: `\everyparagraph` is a macro that is made to look like a token parameter by the use of `\afterassignment`. This latter command sets aside the token `\xevpar` for insertion after the first assignment occurring; then whatever follows is assigned to `\temppar`. After this assignment the macro `\xevpar` unwraps the `\temppar` token list and the constant actions into `\everypar`.

4 Conclusion

In a systematic layout indentation commands need never be typed by the user; they can all be hidden in macros.

Using `\everypar` it is possible to prevent indentation both in single instances, and throughout the document. This has the advantage that it is not necessary to zero the `\parindent` parameter or use `\indent` and `\noindent` instructions.

The approach of employing `\everypar` as sketched above can also be used for a paragraph skip scheme, as I will show in a subsequent article.

References

- [1] Donald Knuth, *The T_EX book*, Addison-Wesley Publishing Company, 1984.
- [2] Leslie Lamport, *L^AT_EX*, a document preparation system, Addison-Wesley Publishing Company, 1986.
- [3] Victor Eijkhout, Unusual paragraph shapes, *Tugboat* vol. 11 (1990) #1, pp. 51–53.
- [4] Stanley Morison, *First principles of typography*, Cambridge University Press, 1936.
- [5] J. Braams, V. Eijkhout, N.A.F.M. Poppelier, The development of national L^AT_EX styles, *TUGboat* vol. 10 (1989) #3, pp. 401–406.