BIJLAGE DD

A parskip scheme

Victor Eijkhout
University of Illinois at Urbana-Champaign
305 Talbot Lab
104 S. Wright Street
Urbana, Illinois 61801, USA
u641001@HNYKUN11

While I was working on the LaTeX styles described in [1], it became apparent to me that lots of people are rather fond of the sort of layout that can be described as

```
\parindent=0cm
\parskip=6pt % or other positive size
```

Unfortunately, most of them realize this layout by no more sophisticated means than simply inserting these two lines at the beginning of the input. The drawback of such a simple action is that all sorts of vertical spaces are augmented by the `\parskip` when there is absolutely no need to, or where it is positively unwanted. Examples of this are the white space below section headings, and the white space above and below list environments in LaTeX.

In this article I will present an approach that unifies the paragraph skip and the white spaces surrounding various environments. Since the macros given below make use of the `\everypar` token list, this article may be seen as a sequel to an earlier paper on an indentation scheme [2], which is based on a similar principle. The `\everypar` parameter was explained there.

\parskip

TeX starts a paragraph when it switches from vertical to horizontal mode. The vertical mode may have been initiated by a `\par` (for instance because of an empty line after a preceding paragraph) or by a vertical skip command; the switch to horizontal mode can be effected by, for example, a character or a horizontal skip command (see the list in [3, p. 283]). Immediately above the first line of the paragraph TeX will then add glue of size `\parskip` to the vertical list[1].

Apparently, then, the `\parskip` parameter is very simple to use. That this is only an apparent simplicity becomes clear in a number of instances.

For instance, unless precautions are taken, the white space below headings is augmented by the paragraph skip. Precautions against this are not particularly elegant; the easiest solution is to include a

```
\vskip-\parskip
```

statement, to backspace the paragraph skip in advance. Such an approach, however, is somewhat error-prone. Vertical spacing will be messed up if what follows is not a paragraph, but a display formula or a box.

Similar considerations apply to the amounts of white space that surround, for example, list environments, as in LaTeX.

Paragraph skip: to be or not to be

(This section is something of a footnote to the rest of the article. Readers who are not interested in layout consideration may skip the rest of it.)

Ordinarily in plain TeX and in LaTeX the paragraph skip is set to `0pt plus 1pt`, which gives pages some 'last resort' stretchability. However, even an amount of vertical space as small as one point may become very visible, and often without need (see for instance the first page of the preface of [3]).

Furthermore, Stanley Morison states that not indenting paragraphs is 'decidedly an abject method' [4]. However, reading his intention instead of his words, he is only concerned with the recognizability of the individual paragraphs. The positive value of the paragraph skip is sufficient to ensure this. If a layout is based on zero values for both `\parindent` and `\parskip`, one may for instance give the `\parfillskip` a positive natural width to prevent last lines of a paragraph from almost, or completely, lining up with the right margin.

Neither Donald Knuth nor Leslie Lamport seem to have given much thought to the case where the paragraph skip has a positive natural width. Leslie Lamport dismisses all potential difficulties with the remark that 'it is customary not to leave any extra space between paragraphs' [5, p. 94].

[1] Unless this paragraph is at the start of a vertical list, for instance at the start of a vertical box or insertion item.

Environments and white lines

Given that the paragraph skip appears to interact with explicit vertical spacing in user macros, it may seem like a good idea to find a unified approach to both. In the rest of this article I will describe the implementation of the following basic idea: *give the paragraph skip the value zero whenever you do an explicit vertical skip*.

For the presentation I assume a context with some form of environments. These are the assumptions that I make about such environments:

- An environment is a portion of material that is vertically separated from whatever is before and after it. Thus, according to this definition, a portion of a paragraph cannot be an environment, nor can an environment start or end in the middle of a paragraph.
- An environment has associated with it three glue parameters: to an environment `foo` correspond `\fooStartskip` (glue above the environment), `\fooParskip` (the paragraph skip inside the environment), and the `\fooEndskip` (glue below the environment).
- At the outset of the environment a
  `\StartEnvironment{foo}`
  statement is executed; at the end of the environment a macro
  `\EndEnvironment{foo}`
  is executed. These statements are assumed to contain a `\begingroup` and `\endgroup` respectively.

Such assumptions are sufficiently general for the macros below to be adaptable to existing macro packages. At first sight it would appear as if section headings are not covered by the above points. However, there is no argument against the start and end of an environment occurring in the same macro.

Tools

First I will present two auxiliary macros: `\csarg` and `\vspace`.

The command `\csarg` is only needed inside other macros; it is meant to enable constructs such as

```
\csarg\vskip{#1Parskip}
```

Its definition is

```
\def\csarg#1#2{\expandafter
    #1\csname#2\endcsname}
```

By way of explanation of this macro, consider a simple example. Let us assume that there exists a macro

```
\def\startlist#1{ ...
    \csarg\vskip{#1Startskip}
    ...}
```

The call

```
\startlist{enumerate}
```

will then lead to the following call to `\csarg`:

```
\csarg\vskip{enumerateStartskip}
```

This expands to

```
\expandafter\vskip
 \csname enumerateStartskip\endcsname
```

Now the `\expandafter` forces `\csname` to be executed before the `\vskip`, so the next step of the expansion looks like

`\vskip\enumerateStartskip` and this statement can simply be executed.

Next I need a generalization of `\vskip`, which I will call `\vspace`: a number of calls to `\vspace` should have the effect that only the maximum argument is placed.

```
\newskip\tempskipa
\def\vspace
 #1{\tempskipa=#1\relax
    \ifvmode \ifdim\tempskipa<\lastskip
        \else \vskip-\lastskip
              \vskip\tempskipa
        \fi
    \else \vskip\tempskipa \fi}
```

This may need some explanation too. First, by the assignment

```
\tempskipa=#1
```

I allow the argument of `\vspace` to be both a control sequence, for instance `\parskip`, and a denotation, for instance `5pt plus 3pt`. If one omits the assignment, the latter option would cause trouble in the `\ifdim` test.

The decision to keep the maximum value of the skip, instead of always replacing the last skip, was motivated by phenomena such as a display formula at the end of a list. If the skip below the display is larger than the vertical glue below the list (which may for instance be zero), the former should be retained entirely.

Note that this macro can insert vertical space of size zero. This will for instance happen if it follows a `\par` command at the end of a paragraph. It is then called in vertical mode, but the last item on the vertical list is a box (the last line of the paragraph) instead of a glue item. The parameter `\lastskip` will then be zero.

The environment macros

In this section, I will give the implementation of the macros `\StartEnvironment` and `\EndEnvironment`.

There is a remarkable similarity between these two macros. As I explained above, the basic idea is to have only explicit spacing above and below the environment; thus, the value of `\parskip` should then be zero both for the first paragraph in the environment, and for the first paragraph that follows it. Both macros should then

- save the current value of the paragraph skip;

- set the paragraph skip to zero;
- give TEX a signal that, somewhere in the near future, the old value of `\parskip` is to be restored.

For this I allocate a skip register and a conditional:

```
\newskip\TempParskip
\newif\ifParskipNeedsRestoring
```

The basic sequence for both macros is then

```
\TempParskip=\parskip
\parskip=0cm\relax
\ParskipNeedsRestoringtrue
```

For both macros, however, this sequence needs to be refined slightly.

The paragraph skip to be 'restored' at the start of the environment is the specific value associated with that environment. This gives us:

```
\def\StartEnvironment
 #1{\csarg\vspace{#1Startskip}
    \begingroup %% make changes local
    \csarg\TempParskip{#1Parskip}
    \parskip=0cm\relax
    \ParskipNeedsRestoringtrue}
```

Note that the statement

```
\csarg\TempParskip{#1parskip}
```

denotes an assignment (without an optional equals sign) here.

At the end of the environment we have to be more careful. It may be the case that the environment being ended was inside another environment, and occurred before the first paragraph inside that environment. In that case the value of `\parskip` is zero, and the proper value must still be restored. Therefore, no further actions are required. We arrive at the following implementation:

```
\def\EndEnvironment
 #1{\csarg\vspace{#1Endskip}
    \endgroup %% restore global values
    \ifParskipNeedsRestoring
    \else \TempParskip=\parskip
          \parskip=0cm\relax
          \ParskipNeedsRestoringtrue
    \fi}
```

Note that both macros start with a vertical skip. This prevents the `\begingroup` and `\endgroup` statements from occurring in a paragraph. On a side note: since these macros are executed in vertical mode, I have not bothered to terminate any lines with comment signs. Any spaces generated by these macros are ignored in vertical mode.

Paragraph skip restoring

So far, I have ignored one important question: how exactly is restoring the paragraph skip implemented. For this I use the `\everypar` token list. Basically then, the idea is the same as in [2]: the occurrence of a paragraph will automatically have TEX perform, through the insertion of the `\everypar` tokens, the actions necessary for subsequent paragraphs.

```
\everypar=
   {\ControlledIndentation
      %see Tugboat 11, #3
    \ControlledParskip}
\def\ControlledParskip
   {\ifParskipNeedsRestoring
       \parskip=\TempParskip
       \ParskipNeedsRestoringfalse
    \fi}
```

The cost of a controlled paragraph skip is then one conditional per paragraph. Conceivably, this cost could even be reduced further (to almost zero) by defining

```
\def\CPS % Controlled Parskip
   {\ifParskipNeedsRestoring
       \parskip=\TempParskip
       \ParskipNeedsRestoringfalse
       \let\ControlledParskip=\relax
    \fi}
```

and including a statement

```
\let\ControlledParskip=\CPS
```

in both the `\StartEnvironment` and `\EndEnvironment` macros, and at the start of the job (for instance by including it in the macro package). This approach, however, does not particularly appeal to me. Too much 'pushing the bit around'.

Conclusion

The `\parskip` parameter is arguably the most tricky parameter of TEX. Its workings are very easy to describe, but in actual practice difficulties arise. In this article I have described how treatment of the paragraph skip can be integrated with the glue above and below environments. As in an earlier article on indentation, I use for this the `\everypar` parameter as an essential tool.

References

[1] J. Braams, V. Eijkhout, N.A.F.M. Poppelier, The development of national LATEX styles, TUGboat vol. 10 (1989) #3, pp. 401–406.

[2] Victor Eijkhout, An indentation scheme, TUGboat vol. 11 (1990) #4.

[3] Donald Knuth, The TEX book, Addison-Wesley Publishing Company, 1984.

[4] Stanley Morison, First principles of typography, Cambridge University Press, 1936.

[5] Leslie Lamport, LATEX, a document preparation system, Addison-Wesley Publishing Company, 1986.