Stubborn leaders and juggling boxes: A slightly unusual table of contents

Boris Veytsman

Abstract

A macro for typesetting an unusual table of contents is introduced. The history of development of this macro is described in detail showing the refinement of requirements and refactoring of the code. The TEX boxes, their measuring and inspection are discussed.

1 Introduction

Many publishers that accept manuscripts in TEX or LATEX put their styles on CTAN; see the (certainly incomplete) list at `http://www.ctan.org/keyword/publishers`. However, when a journal uses TEX internally, there are also macros not publicly released. These macros are used in typesetting covers, journal title pages, issue and volume tables of contents, indices etc. They are specific to the journal. Without major changes they can be used only by the people who want to create another one with exactly the same look and feel. For some reason this practice is frowned upon by the publishers. Still, sometimes the tricks in these macros might be of interest to TEXnicians. Thus extracting them and publishing separately seems to be worth doing.

I recently released the package (Veytsman, 2012) for typesetting articles for the journal *Res Philosophica*, `http://www.resphilosophica.org`. I also wrote a package to typeset the covers and technical information for the journal. In this article I describe a somewhat non-trivial macro used in the table of contents of the journal.

An important (maybe *the* most important) part of the work of a programmer is the understanding of the requirements. Thus I would like to tell the history of this macro: how it was designed and re-designed after studying the samples.

2 The problem and a naïve solution

The entries of the journal's table of contents were typeset like the ones in Figure 1: the article title, then the dots connecting the entry to the page number, following by the page number itself, and the authors on the next line in italics. The dots are called *leaders* in TEX. LATEX has a nice macro `\dotfill` which makes infinitely stretchable leaders.[1] With this macro our TEX source looks like this:

[1] While most of our code works in plain TEX, we do use some LATEXisms like `\dotfill`, `\itshape`, etc., which are easy to emulate in other formats.

Article title 5
A. U. Thor

Figure 1: A simple TOC entry

Article title 5
A. U. Thor

A very very very very very very long article title 9
A. N. Other

Figure 2: Failure of the naïve solution

```
% #1 is the title, #2 is the author(s)
% #3 is the page number
\def\settocentry#1#2#3{\par
  #1\space\dotfill\space#3\par
  \textit{#2}\par\medskip}
\parskip=0pt\parindent=0pt
\settocentry{Article title}{A. U. Thor}{5}
```

The problem is solved!

Unfortunately, it is not so simple. What happens if the article title is very long? The result is shown in Figure 2. The leaders start from the second line of the title, separating it from the author, which just does not look right. This situation occurs not infrequently because the width of the journal pages is rather small (6 in.).

An examination of the samples showed that the designer wanted (1) the title and the authors to occupy no more than 60% of the text width, and (2) the leaders to *always* start from the first line of the entry, even if the title occupied two or more lines. Of course, there are some arguments against this design. One can object to the fact that the title is "broken" by the leaders. On the other hand, the eye looking for a title or authors moves in the vertical direction and probably disregards the intervening leaders, so this objection might not be too strong after all. Anyway, this was the design which I was given to implement.

3 A less naïve solution

A less naïve idea is the following. Let us typeset the title in a box of certain width, say `\entrywidth`, with the baseline at the first line. If it occupies one line, we throw away the box and use the solution from the previous section. However, if it occupies two or more lines (which we can check by measuring its depth), we align it with the leaders and page number.

To make our code simpler, let us split the main macro into two parts: setting the title and setting the authors. We also put the page into a global

Article title 5
A. U. Thor

A very very very very very very long article title 9
A. N. Other

Figure 3: A less naïve solution

`\articlepage` to be used by the `\setarticletitle` macro — mostly to avoid cluttering the definitions:

```
\def\settocentry#1#2#3{%
  \gdef\articlepage{#3}%
  \setarticletitle{#1}\par
  \setarticleauthors{#2}\par\medskip}
```

To set the article title we put it in a `\vtop` box `\titlebox`. Since it is `\vtop`, the baseline of the box coincides with the baseline of the first line, and to check the presence of the second line we need to measure its *depth* rather than height. Also, we add `\strut` at the end of the box to keep the distance between the lines uniform:

```
\newdimen\tocboxwidth
\tocboxwidth=0.6\textwidth
\newbox\titlebox
\def\setarticletitle#1{\par
  \setbox\titlebox=\vtop{%
    \hsize=\tocboxwidth\relax
    #1\strut\par}%
  \ifdim\dp\titlebox<\baselineskip\relax
  % the box is one line long, forget it
    #1\space\dotfill\space\articlepage
  \else % The box has more than one line
    \vtop{\hsize=\textwidth\leavevmode
      \box\titlebox\space\dotfill
      \space\articlepage}%
  \fi}
```

We also set the authors in a box of the same width:

```
\def\setarticleauthors#1{%
  \vtop{\hsize=\tocboxwidth
    \strut\itshape#1\strut}}
```

Is our problem solved? Unfortunately not. First, titles are usually no more than two lines long. Therefore it would be better to typeset them ragged right. Second, often the article title is split at the logical break, for example, after a colon. Therefore we can expect an input like this:

```
\parskip=0pt\parindent=0pt\raggedright
\settocentry{Article title}{A. U. Thor}{5}
\settocentry{A very very very very
  longwinding article
  title}{A. N. Other}{9}
\settocentry{The state of the art:\\
  New developments and results}%
  {C. O. R. Respondent}{21}
```

Article title 5
A. U. Thor

A very very very very longwinding article title 9
A. N. Other

The state of the art: New developments and results 21
C. O. R. Respondent

Figure 4: Failure of the less naïve solution

As seen in Figure 4, this input breaks our TOC. The leaders start at the end of the box, leaving an ugly gap. Note that if not for the leaders, our solution would be perfect: the page numbers are aligned with the first lines of the titles.

4 The current solution

Whenever I hit a wall in my TEX hacking, I try to find a clue in the book by Eijkhout (2007). This worked for me again. In §5.9.6 there I found a nice trick using `\lastbox`. This command deletes the last box from the vertical list and makes it available for inspection and retypesetting. If we apply this command iteratively, we "pick apart" the list line by line until we hit the top. Eijkhout used it to typeset the lines in a paragraph in a way that depended on their natural width. He put the paragraph in a box, and then inspected each line starting from the last one, re-typesetting if necessary (see also Eijkhout, 1990).

This trick provides the solution to our problem. Indeed, let us typeset the title in a box of width `\tocboxwidth`. Then let us pick this box apart, moving the lines into another box. When we hit the last line from the bottom (i.e. the first line from the top), we break this line and re-typeset it with the leaders and the page number.

There are two additional problems to be solved. First, a vertical list has not just boxes, but also glue and penalties. Eijkhout found that he needs to put `\unskip` and `\unpenalty` in his loop after getting the last box. LATEX `\raggedright` adds additional skips to the lines, so after some meditation over the kernel code and trials I put there a longer incantation with one `\unpenalty` and three `\unskip`s. Second, how do we know that we have hit the top line? The only way to find it is on the *next* iteration of the loop, where `\lastbox` returns an empty

box. Thus we need to postpone moving the line to the result until we inspect the next one. Therefore the final algorithm includes four boxes: `\trialbox` to initially typeset the title, `\resultbox` to put the result in, `\lastlinebox` for the line read last, and `\prevlinebox` for the previously read line. The algorithm is the following:

1. We typeset the title into a `\trialbox`.
2. On each iteration of the loop we take the last line of this box and put it in the box `\lastlinebox`.
3. If this box is empty, we have hit the top. Then we break `\prevlinebox`, add leaders and page number and put on the top of `\resultbox`.
4. Otherwise we put `\prevlinebox` in `\resultbox`, put `\lastlinebox` in `\prevlinebox` and repeat the loop.

A final note before we start going into the code: when we add vertical boxes, we need to check first whether they are empty, otherwise we introduce unwanted vertical space.

The `\setarticle` macro creates `\trialbox` and then typesets the `\resultbox`:

```
\newbox\trialbox
\newbox\resultbox
\newbox\lastlinebox
\newbox\prevlinebox
\def\setarticletitle#1{%
  \setbox\trialbox=\vtop\bgroup
    \hsize=\tocboxwidth\relax
  \strut#1\strut\par\getlastline\egroup
  \box\resultbox}
```

The heart of our algorithm is the `\getlastline` macro. It implements iteration in a very TEXish way: by recursively calling itself.

```
\def\getlastline{%
  % Reading the last line
  \global\setbox\lastlinebox=\lastbox
  \ifvoid\lastlinebox % We hit the top;
                      % construct the
                      % result and
                      % finish
    \global\setbox\resultbox=\vtop
      \bgroup\hsize=\textwidth
        \hbox to \textwidth
        {\unhbox\prevlinebox
          \unskip\unpenalty\space
          \dotfill\space\articlepage}%
        \ifvoid\resultbox\else
        \box\resultbox\fi
      \egroup
  \else % We did not hit the top yet
    \unskip\unpenalty\unskip\unskip
```

Article title 5
A. U. Thor

A very very very very 9
longwinding article title
A. N. Other

The state of the art: 21
New developments and
results
C. O. R. Respondent

Figure 5: The current solution

```
    \ifvoid\prevlinebox\else
      \global\setbox\resultbox=\vtop
      \bgroup
        \box\prevlinebox
         \ifvoid\resultbox\else
         \box\resultbox\fi
      \egroup
    \fi
    \global\setbox\prevlinebox
      \box\lastlinebox
    {\getlastline}% Recursion!
  \fi}
```

The results are shown in Figure 5. As one can see, now the leaders behave correctly.

5 Conclusion

Our article shows how TEX boxes provide a rich environment for a rather complex typesetting problem. The `\lastbox` command is a powerful tool which can be used for many non-trivial tasks.

References

Eijkhout, Victor. "Unusual Paragraph Shapes". *TUGboat* **11**, 51–53, 1990. http://www.tug.org/TUGboat/tb11-1/tb27eijkhout.pdf.

Eijkhout, Victor. *TEX by Topic*. Lulu, 2007. http://eijkhout.net/texbytopic/texbytopic.html.

Veytsman, Boris. *Typesetting Articles for* Res Philosophica, 2012. http://mirror.ctan.org/macros/latex/contrib/resphilosophica.

◇ Boris Veytsman
School of Systems Biology &
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net
http://borisv.lk.net