Updating the nostarch class
Boris Veytsman

Abstract
No Starch Press’s house style has interesting typographic features. Their implementation posed some \TeXnical challenges described in this paper.

1 Introduction
No Starch Press was founded in 1994 by a charismatic publisher, Bill Pollock, who proudly displays on the company’s web page a personal story of being hired — and fired — by the major players in the business until he established his own (see nostarch.com/about). No Starch Press positions its products as the finest in geek entertainment and boasts such titles as Python Crash Course, Python for Kids, How Linux Works, and Hacking: The Art of Exploitation, with topics spanning security, hacking, LEGO, and series like The Manga Guide, covering Biochemistry, Calculus, Cryptography, etc., up to the Universe.

It is not surprising that this publisher has strong opinions about the design of their books. Indeed, No Starch Press books have a distinct house style, sometimes rather unusual for technical literature. It was an interesting challenge to implement them in \LaTeX. I started this task in 2008 with the first release of the nostarch class. Since then the publisher’s team made many ad hoc changes to the original class, suitable for the tasks at hand. Some of these changes assumed manual adjustments of the input, which was error-prone and time consuming. The advent of Overleaf increased the number of authors and added new features. In this paper I describe several typographical challenges presented by the class and the ways I approached them.

2 URL breaking
Since the advent of the Internet, URLs have been a nemesis of compositors. A long string of text with no obvious hyphenation points presents an obvious problem for paragraph setting.

The package url [1] solves this problem by allowing breaking of URLs. It is a highly customizable package, used internally by the ubiquitous hyperref [4], and is the de facto standard of URL typesetting in the \LaTeX world.

Still, even with this package certain problems remain. For example, how do we split a URL containing a hyphen? If we split \url{https://hyphenated-url.org} before \url{url.org} (as happened here!), the reader may be confused as to whether the hyphen belongs to the URL or just signifies the break. That is why the url package does not allow breaking after hyphens by default: you need to explicitly enable it with the hypen option. No Starch Press offers a rather elegant solution: it allows breaks before hyphens only, so the reader is not confused by the trailing hyphens.

The full house rules are more complex. Breaks are allowed as follows.

1. After the symbols: \# > ] ) } :
2. Before the symbols: . _ = & - ! ? , @ ' ” + < [ ( { ; } ] ) ”
3. Breaks are not allowed before /, so expressions like \url{https://} are never split.

To implement these rules, we first note that the url package typesets URLs as math expressions. The symbols after which breaks are allowed are declared as mathop or mathbin atoms, and the standard \LaTeX rules for inline math expressions line breaking apply. However, this approach needs a modification if we want to break an expression before a symbol. Fortunately, the package provides a \UrlSpecials macro, which allows one to associate any behavior with any symbol. Thus we can allow breaks before the dot using the following code:

\begin{verbatim}
\g@addto@macro{\UrlSpecials}{% \\
\do\./{\penalty\UrlBreakPenalty\mathchar46\relax}}
\end{verbatim}

Note that \mathchar46 is a dot symbol. Similarly, we can disallow any break before the slash with:

\begin{verbatim}
\g@addto@macro{\UrlSpecials}{% \\
\do\/{\unpenalty\penalty\UrlBreakPenalty}}
\end{verbatim}

There is an additional problem if the author uses the amsmath [2] package. An attempt to redefine the opening bracket leads to the Bad mathchar (32768) error message. At TUG’23 David Carlisle explained that the problem is in the macro \resetMathstrut\relax added by amsmath to the \everymath hook and redefining the character code of the opening bracket. Fortunately, the url package has its own hook, \UrlMathSetup, called after \everymath. Thus we can nullify this macro for URLs only:

\begin{verbatim}
\g@addto@macro{\UrlMathSetup}{% \\
\let\resetMathstrut\relax}
\end{verbatim}

3 Chapter opening
The chapters in No Starch Press books have quite an impressive opening: the first paragraph is typeset in a larger font, and there is a space for the “circular art”, as shown in Figure 1. Happily, \LaTeX has

Boris Veytsman
doi.org/10.47397/tb/44-2/tb137veytsman-nostarch
3

CURABITUR VITAE LECTUS SIT amet turpis pretium condimentum


Figure 1: A chapter opening in No Starch Press style


We just need to calculate this width automatically.

The situation with tables is a little bit more complex. When we typeset the caption of a table, we do not know the width of the table, since the body is not yet typeset. Therefore we use the usual \TeX trick: at the end of a table we save the width of the body to the .aux file. At the beginning of the table we check if it has been defined in the previous file. At the beginning of the sectioning macro, the nostarch class adds to this hook, among other things, the following code, which uses the switch \ifnostarch@firstpara:

\everypar{%
  \ifnostarch@firstpara
    \rule{\z@}{49\p@}%
    \ifx@chapterart\empty
      \else
        \makebox[0pt][r]{%
          \raisebox{-0.5in}[0pt][0pt]{%
            \chapterart\hspace(0.21in)}%}
        \parshape=5
        0.963in 3.622in 0.913in 3.672in
        0.813in 3.772in 0.563in 4.022in
        0in \textwidth
        \fi
        \fontsize{16pt}{16.5pt}\selectfont\parskip=3pt
      \else
        \parskip=Opt \normalsize\selectfont
        \fi
      \@firstparafalse}
\fi
\ifnostarch@measurecaptionwidth
  \newcommand{\nostarch@measurecaptionwidth}(\%\ni@newbox{\nostarch@captionwidth}{\wd\@tempboxa}\
\else
  \ifnostarch@overridecaptionwidth
    \nostarch@captionwidth\%\fi
\fi
\par
\setbox\@tempboxa\hbox{\unbox\@tempboxa}\
\global{\setlength{\nostarch@captionwidth}{\wd\@tempboxa}\
  \box\@tempboxa\par
  \global{\nostarch@overridecaptionwidthfalse}}

The flag \ifnostarch@overridecaptionwidth is discussed below.

Figure 2: A figure in No Starch Press style

A better solution would be to use the paragraph hooks relatively recently added to the \LaTeX kernel.

4 Captions

The most difficult task so far has been automatic formatting of captions. No Starch Press house style does not center figures and tables: they are left justified with figure captions after the figures, and table captions before the figures. However, a full width caption with a narrow left justified figure body looks rather ugly. Therefore the house style has another requirement: the width of the caption should be no longer than the width of the body, as demonstrated in Figures 2 and 3. Fortunately, the \texttt{caption} package [5], used by the \texttt{nostarch} class internally, allows typesetting a caption in a parbox of the given width. We just need to calculate this width automatically.

It is relatively easy to do with figures. In most cases figures have just one graphical box. We measure this box and use it to typeset the width of the caption. Note that in \LaTeX the examination of the last box is a destroying operation: we need to return the box to the list if we want to preserve it. We add to the \texttt{endfigure} the following command,

\nostarch@measurecaptionwidth, defined as:

\newcommand{\nostarch@measurecaptionwidth}(\%\ni@newbox{\nostarch@captionwidth}{\wd\@tempboxa}\
\else
  \ifnostarch@overridecaptionwidth
    \nostarch@captionwidth\%\fi
\fi
\par
\setbox\@tempboxa\hbox{\unbox\@tempboxa}\
\global{\setlength{\nostarch@captionwidth}{\wd\@tempboxa}\
  \box\@tempboxa\par
  \global{\nostarch@overridecaptionwidthfalse}}

The flag \ifnostarch@overridecaptionwidth is discussed below.

The situation with tables is a little bit more complex. When we typeset the caption of a table, we do not know the width of the table, since the body is not yet typeset. Therefore we use the usual \LaTeX trick: at the end of a table we save the width of the body to the .aux file. At the beginning of the table we check if it has been defined in the previous
Another special case is long tables typeset with the `longtable` package [3]. Fortunately, long tables know its width, so we just need to read it. Here is the corresponding code:

```latex
\ifnostarch@overridecaptionwidth\else
  \global\setlength{\nostarch@captionwidth}{\z@}\% 
\bgroup 
  \def\LT@entry##1##2{\global\addtolength{\nostarch@captionwidth}{##2}}% 
  \csname nostarch@captionwidth\endcsname
\csname LT@\romannumeral\c@LT@tables\endcsname
\egroup
\fi
\global\nostarch@overridecaptionwidthfalse
```

Above, we deferred the discussion of the flag `\ifnostarch@overridecaptionwidth`. In fact, no automatic system is 100% accurate. Sometimes a figure or a table contains several boxes of different widths arranged vertically. Sometimes it is too narrow. Thus it makes sense to allow the user to override the algorithm. The command `\NextCaptionWidth` with one argument does just that:

```latex
\newcommand\NextCaptionWidth[1]{\global\nostarch@overridecaptionwidthtrue\csname LT@\romannumeral\c@LT@tables\endcsname}
```

It sets the width of the following caption to the argument and informs the measuring code to skip the measuring.

5 Conclusion

The \TeX programming layer, even in the “old” incarnation of \TeX{} 2\textasciitilde, is quite flexible. It can satisfy many typesetting requirements and provide automatic composition—even when the requirements are rather unusual, as with some of those of No Starch Press.

References

[2] \TeX{} Project Team. The `amsmath` package, 2023. ctan.org/pkg/amslatex

\diamond Boris Veytsman

borisv (at) lk dot net
https://borisv.lk.net