First Line Special Handling with TEX

Anne Brüggemann-Klein
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
Postfach 6980
7500 Karlsruhe, West Germany

Jim Sterken in TUGboat 4, no. 2, and James Alexander in TUGboat 7, no. 2, ask a question about fancy first line processing: Can TEX automatically set the first line of a paragraph in a special font? Yes, indeed, TEX can! But before I tell you a solution, let me describe some wrong ways I tried before, because this explains some assumptions and restrictions I made. For short, I call paragraphs which have their first lines set in a special font, *special* paragraphs, in contrast to the *normal* ones.

The first line problem in its strongest form means the following: Can TEX make *optimal* special paragraphs in the same sense as it does optimal line breaking for normal paragraphs? In my opinion, the only solution for this strong form is to alter TEX's line breaking algorithm, but this is strictly opposite to the spirit of the TEX community.

Therefore I give a somewhat weaker formulation of the first line problem: Can TEX find an acceptable breakpoint for the first line of a special paragraph and then do optimal linebreaking as usual for the rest of it? The question is now: How can you find an acceptable breakpoint for the first line of the paragraph?

My first idea was to impose this job on TEX's line breaking algorithm: First let TEX set the whole paragraph in the special font for the first line and store it in a box register. Then take the first line of this box as the first line of the paragrah, "unbox" the rest of the box and set it again, but in the normal font.

This nice idea didn't work, and the reason is that TEX's digestive process is a one-way street. There is no problem to set the whole paragraph in the special first line font and store it in a box register, and you can easily detract the first line from this box by a `\vsplit`-command. It's somewhat harder to "unbox" the rest of the box by the help of the commands `\vsplit`, `\lastbox`, and `\unhbox`. But even suppose you can manage this, all what you get is a list of character boxes, and there is no way to change this back into a token list in order to set the stuff a second time, but in another font.

So I had to help TEX in finding the breakpoint for the first line. My idea was to put one word after another into an `\hbox`, using the special font for the first line, until — after some stretching or shrinking — the width of this box matches the `\hsize`. My intention was to regulate the amount of the stretching or shrinking by testing the "badness" of the according line. But unfortunately the badness of a line is only *reported* in your log-file. You cannot *use* it as an internal parameter inside of TEX.

Therefore, I had to calculate explicitly whether the box fits into a line by itself. For this purpose, I had to calculate the total stretchablility and shrinkability of the box. By the help of `\sfactor` and `fontdimen3` and `4` this can be done.

But I decided to make things easier. To me it seems not necessary to do tricky spacing in a heading-like special first line. Therefore I set the first line in a `\frenchspacing` style. Then I only had to count the spaces in the box and multiply `\fontdimen3` and 4 of the first line font by this factor to obtain the stretchability and shrinkability of this box.

Now the algorithm which finds an appropriate breakpoint for the first line works as follows. One word after the other is appended to a box until a feasible breakpoint is found. When a word has been appended, a test is made whether the box fills a line: If the width of the box is smaller than the `\hsize`, we have two cases: In the first case the box cannot be stretched to the `\hsize`. In this case the box is too short, i.e. another word must be appended and the process is iterated. In the second case, when the box can be stretched to the `\hsize`, this box will be the first line of the paragraph. In this case the glue set ratio is less or equal 1, i.e. the badness of the line is less or equal 100.

If, on the other hand, the width of the box has become greater than the `\hsize`, there are two cases again: In the first case, the box can shrink to the `\hsize`, and again we have a badness less or equal 100 if we take this box as the first line of the paragraph. In the second case the box is too large to shrink to the `\hsize`. In this case, the last word of the box is removed, and the rest of the box is my first line. Then the badness of the line is greater than 100, and it might be an underfull one.

The algorithm prefers to stretch the first line instead of shrinking it, which fits well to the heading-like character of the application. It seems better to me to stretch the first line than to hyphenate the last word of it (hyphenation could be included, but the user had to indicate possible hyphenation points in the input). Furthermore, there is no problem to multiply the stretchability and shrinkability of the first line by a factor to allow for a higher badness.

Some features of TEX don't work in the context of the macros described here. One of these is migrating material from the first line (`\vadjust`, e.g.). Special caution is necessary with macros. Spaces inside of macros and their arguments must be protected by grouping symbols. Some macros, for example the `\verbatim`-macro from the TEXbook, don't work at all near the beginning of a special paragraph, because the token converting process is disturbed. Finally, my macros don't work if the paragraph doesn't take at least two lines of text.

The "user interface" of the macros is very simple. You can use the commands `\firstlinefont{<filename>}` to define the font for the first line of a special paragraph (default: `\firstlinefont{cmcsc10}`) and `\firstlineindent{<dimen>}` to define its indentation (default: `\firstlineindent{0pt}`). With `\iffirstlineinner` you can test whether you are in the first line of a special paragraph or not, and `\firstlinespecial` in vertical mode starts a special paragraph. `\firstlinespecial` works also inside an `\everypar`-command, but be shure to remove the indentation box first.

If you are a mathematician (like me), these macros can add a playful or narrative element to your severe subject. But if you are a philologist, why don't you improve the aesthetic effect by combining fancy first line printing with fancy printing of initials?

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                         %%%%%
%%%%%           FIRST LINE SPECIAL HANDLING WITH TeX          %%%%%
%%%%%                                                         %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These macros can set the first line of a paragraph
% in a special font. You can determine this font by the
% command \firstlinefont{<filename>}.
% The default value is \firstlinefont{cmcsc10}.
% You can define the indentation of such a paragraph by
% the command \firstlineindent{<dimen>}. The default value is
% \firstlineindent{0pt}.
% There is a if-register \iffirstlineinner, which is true if and
% only if you are in the first line of a special paragraph.
% Then start the paragraph with the command \firstlinespecial
% in vertical mode.

\catcode`\@=11 % `@' is used as a letter to hide command names
               % from the user.

%%% Allocate some internal registers:
%%% The box \oldb@x contains the material which has been
%%% collected for the first line so far.
%%% \newb@x tries to append another
%%% word to \oldb@x.
%%% The counter \spacec@unt counts the number of interword spaces
%%% in \oldb@x.
%%% The dimen-register \firstlineind@nt stores the indentation of
%%% the special paragraphs.
%%% The if-register \iffirstw@rd stores whether the first word
%%% of the special line has been read already or not.
%%% The dimen-registers \sp@cestretch and \sp@ceshrink store
%%% the strechability and the shrinkability of the normal interword
%%% space in the font \firstlinef@nt.
%%% The dimen-register \boxw@dth stores the width of the box \newb@x.
%%% The if-register \iffirstlineinner is true iff you are in the
%%% first line of a special paragraph.

\newbox\oldb@x
\newbox\newb@x
\newcount\spacec@unt
\newdimen\firstlineind@nt
\newif\iffirstw@rd
\newdimen\sp@cestretch
\newdimen\sp@ceshrink
\newdimen\boxw@dth
```

```
\newif\iffirstlineinner \firstlineinnerfalse

%%% \firstlinefont determines which font has to be used
%%% for the first line of a paragraph.
%%% This font gets the internal name \firstlinef@nt.
%%% Default is \firstlinefont{cmcsc10}.
%%% \firstlineindent defines the indentation of a paragraph which
%%% has been started with \firstlinespecial.
%%% This dimension is stored in the register \firstlineind@nt.
%%% Default is \firstlineindent{0pt}.

\def\firstlinefont#1{\font\firstlinef@nt#1}
\firstlinefont{cmcsc10}
\def\firstlineindent#1{\firstlineind@nt #1}
\firstlineindent{0pt}

%%% \firstlinespecial initializes some registers, stores the current
%%% font, switches to \firstlinef@nt,
%%% and starts the command \n@xt which does the real work.

\def\firstlinespecial{%
     \firstlineinnertrue
     \setbox\oldb@x\hbox{\hskip\firstlineind@nt}%
     \setbox\newb@x\copy\oldb@x
     \spacec@unt 0
     \firstw@rdtrue
     \xdef\oldf@nt{\the\font}%
     \firstlinef@nt
     \sp@cestretch\fontdimen3\firstlinef@nt
     \sp@ceshrink\fontdimen4\firstlinef@nt
     \n@xt}

%%% The command \append copies \oldb@x to \newb@x and appends
%%% its argument to \newb@x, preceeded by a space, depending on
%%% \iffirstw@rd.

\def\app@nd#1{%
     \setbox\newb@x\hbox{\unhcopy\oldb@x
     \iffirstw@rd\else\space\global\advance\spacec@unt by 1\fi
     #1}\firstw@rdfalse}

%%% The command \n@@xt is the heart of the whole. At the
%%% beginning of a \firstlinespecial-command, \n@xt has the
%%% same meaning as \n@@xt. When \n@@xt comes to work, \oldb@x
%%% contains the material which has been collected so far for the
%%% first line of the paragraph, i.e.~\oldb@x is an initial part of
%%% the first line.
%%% \n@@xt reads the next word of the paragraph and appends it to
%%% \newb@x, which is a copy of \oldb@x. As its last command, \n@@xt
%%% calls in \n@xt again, which iterates the process.
```

```
%%% But how to stop the process? After \n@@xt has
%%% appended a word to \newb@x, it makes a test: If the
%%% width of \newb@x is still less than \hsize we have two cases:
%%% In the first one the line is still too short and the process must
%%% be iterated. In the second one, the interword glue of the box
%%% can be stretched to match \hsize. In this case we can take
%%% \newb@x as the first line of the paragraph, and \n@xt must be
%%% redefined to stop the process.
%%% On the other hand, if the width of \newb@x has become greater or
%%% equal to \hsize, we can try to shrink \newb@x to \hsize.
%%% If \newb@x doesn't shrink to
%%% \hsize, we have to remove the last word from \newb@x. In fact, in
%%% this case we return to \oldb@x and take it as the first line.
%%% Here you may get a message about an underfull \hbox. Then
%%% the second line starts with the last word and \n@xt is
%%% redefined as before.

\def\n@@xt#1 {% Note that the argument is delimited by a blank!
     \app@nd{#1}%
     \boxw@dth\wd\newb@x
     \ifdim\boxw@dth < \hsize
           \advance\boxw@dth by \spacec@unt\sp@cestretch
           \ifdim\boxw@dth < \hsize
                 % \message{1 }% not yet full enough
                 \else
                 \message{2 }% stretchable to \hsize
                 \noindent\line{\unhbox\newb@x}\penalty -10000
                 \def\n@xt{\firstlineinnerfalse\let\n@xt\n@@xt}%
                 \oldf@nt
                 \fi
           \else
           \advance\boxw@dth by -\spacec@unt\sp@ceshrink
           \ifdim\boxw@dth > \hsize
                 \message{3 }% last word in next line, shrinkable
                 \noindent\line{\unhbox\oldb@x}\penalty -10000
                 \def\n@xt{\firstlineinnerfalse\let\n@xt\n@@xt}%
                 \oldf@nt
                 #1 % keep this blank!
                 \else
                 \message{4 }% shrinkable to hsize
                 \noindent\line{\unhbox\newb@x}\penalty -10000
                 \def\n@xt{\firstlineinnerfalse\let\n@xt\n@@xt}%
                 \oldf@nt
                 \fi
           \fi
     \setbox\oldb@x\box\newb@x
     \n@xt}
\let\n@xt\n@@xt
\catcode`\@=12
```