The PracTEX Journal
TPJ 2005 No 04, 2005–10–22
Rev. 2005–10–31

A Brochure

M. A. Guravage

I had the occasion recently to read a particular company's brochure and, perhaps this is a curse on people who use TEX for too long, I noticed how poorly typeset it was. I suspect someone had composed it with a page-layout program; dragging blocks of text around with little regard for either margins or alignment. This was the impetus for me to write a short example describing the necessary relationship in a brochure between columns and margins, and to demonstrate how simply this can be done with ConTEXt.

A brochure is a document, typeset on a single oblong piece of paper, whose text is set in columns such that each column becomes a separate entity when the paper is folded. The example that followes is a tri-fold brochure, i.e. three columns two folds, typeset in landscape orientation on A4 paper. Not surprisingly, we begin by specifying the paper size and orientation, centering the typeset page, and turning off page numbers.

```
\setuppapersize [A4,landscape] [A4,landscape]
\setuplayout [width=middle]
\setuppagenumbering [state=off]
```

Snippet 1 preliminaries

Next we specify that the brochure will have three columns. ConTEXt offers two ways to place text in columns: the original column commands, and the newer columnset mechanism. Though either will do, columnsets are more flexible. You can read more about columnsets in the *Columns* **manual**.

Now we add some content. When **snippet 3** is typeset, ConTEXt chooses a distance between the columns that, while appropriate for columns on an open page, is too small for a folded brochure.

```
\definecolumnset [brochure] [n=3]
```

Snippet 2 columnset definition

```
% interface=en output=pdftex
\setuppapersize [A4,landscape] [A4,landscape]
\setuplayout [width=middle]
\setuppagenumbering [state=off]
\definecolumnset [brochure] [n=3]
\starttext
  \startcolumnset[brochure]
    \dorecurse {6} {
      \subject{Column \recurselevel}
        \processfile{knuth}
       \column}
  \stopcolumnset
\stoptext
```

Snippet 3 inter-column space too small

If we were to print and fold **snippet 3**, we would see that with ConTEXt's default page layout, the columns are not centered. To ensure that the space on either side of each column is the same when the brochure is folded, the inter-column spacing must be twice that of the page's margin. In ConTEXt margins are controlled by the `backspace` dimension. We make the inter-column spacing dependent on the margins by assigning the `distance` dimension when we define the columnset.

```
\definecolumnset [brochure] [n=3, distance=2\backspace]
```

Snippet 4 set inter-column distance

Snippet 4 positions the columns correctly relative to the page and one another but, unfortunately, the default `backspace` dimension of $2\frac{1}{2}$ centimeters is too wide. We can control the size of the margins, and by implication the width of the columns' text areas, by adjusting the `backspace` in the `setuplayout` command as follows:

```
\setuplayout [width=middle, backspace=1cm]
```

Snippet 5 explicit backspace

While it is sufficient to specify the number of columns and `backspace` dimension explicitely, their relationship is highlighted if we make the margin width dependent on the number of columns and as a proportion of the page width. Instead of saying that our brochure has three columns and the `backspace` is one centimeter, we say, rather, that the brochure has three columns and that the `backspace` is derived such that the margins will occupy a quarter of the page width.

I prefer proportions over explicit dimensions. **Snippet 6** shows how to set ConTEXt's `backspace` dimension as a function of the number of columns and the margins' percentage of the page width. Use the `numberofcolumns` and `percentagewhitespace` macros to set your preferences, and the `margin` macro will calculate the corresponding `backspace` dimension.

```
\def\numberofcolumns{3} % number of columns
\def\percentagewhitespace{4} % proportion of the page
\def\margin{\dimexpr(\dimexpr(\paperwidth/\percentagewhitespace)/
                          (\numberofcolumns * 2))}
\setuplayout [width=middle, backspace=\margin]
\definecolumnset [brochure] [n=\numberofcolumns, distance=2\backspace]
```

Snippet 6 implicit backspace

With the columns typeset correctly, we can turn our attention to the column headings. In the example I topped each column with a `subject` command. But to reinforce the column motif, I want to change the default section title appearance so that it is centered above the column on a colored background that spans the column's width. This is achieved by

defining a macro named `mysubject` that redefines the default behavior of the `subject` command.

```
\setupcolors [state=start]
\def\mysubject#1#2%
  {\framed [frame=off, offset=0.5em,
            backgroundcolor=lightgray, background=color,
            width=broad]{\midaligned{\CapStretch{\sc #1}}}}
\setuphead [subject] [command=\mysubject]
```

Snippet 7 custom subject title

We should pay attention to the order in which the columns are printed so that thay appear in the correct order when the brochure is folded. The traditional sequence for a tri-fold is columns 2, 3, and 4 on one side and columns 5, 6 and 1 on the other. With columns 2, 3 and 4 face up, fold column 4 to the left and column 2 to right; which positions column 1 as the front of the brochure.

The example uses a `dorecurse` loop to create the correct number of columns. In practice you would set each column separately, for example:

```
\startcolumnset[brochure]
  \subject{Column 2} \processfile{tufte} \column
  \subject{Column 3} \processfile{zapf}  \column
  \subject{Column 4} \processfile{ward}  \column
  \subject{Column 5} \processfile{knuth} \column
  \subject{Column 6} \processfile{bryson}\column
  \subject{Column 1} \processfile{davis} \column
\stopcolumnset
```

Snippet 8 typeset columns separately

Finally, when you look through the example you will notice two pieces of code that have nothing to do with brochures per se, but that are quite interesting in their own rights. I encourage you to look at them. The first is the `fakewords` command; it generates the

fake text used to populate the columns. You can read about faking text in the *This Way* magazine *Faking Text and More*. The second is the `stretchednormalcase` command; it increases the spacing between letters and is the basis for the `CapStretch` macro used to redefine the default appearance of `subject` titles.

Bringing these bits and pieces of code together, the accompanying complete example allow you to try various combinations of column counts and margins proportions quickly and easily. Like other recipes the example contains the essentials, and is a useful means to a variety of ends.