The parcolumns package*

Jonathan Sauer
jonathan.sauer@gmx.de

2004/11/25

Abstract

This file describes the parcolumns package that provides an environment for typesetting text in two or more columns columns in parallel.

Contents

1 Introduction

Sometimes it is necessary to typeset text in two or more columns in parallel, i.e. when typesetting a text in its original language and in its translation(s). This package provides the `parcolumns` environment for typesetting text in several columns, where the text of all columns is aligned vertically.

Text in the `parcolums` environment is typeset in chunks, where the chunks of one row are aligned vertically. A chunk consists of one or more paragraphs of text, including TEX macros.

*This document corresponds to parcolumns.sty v1.2, dated 2004/11/25.

2 Usage

`parcolumns` Usage: `parcolumns [`⟨*options*⟩`] {`⟨*number of colums*⟩`}`.

Inside the `parcolumns` environment text can be typeset in two or more columns in parallel via the `\parchunk` macro. Normal text can also be included.

The `parcolumns` environment takes an optional parameter that specifies the options for the environment using the keyval and processkv packages. The following options exist:

colwidths Sets the widths of the columns. The widths are specified as key-values, ⟨*columnnumber*⟩=⟨*width*⟩. Columns start at '1'. Note that in order to be ignored by the keyval package, the complete value has to be surrounded by braces, i.e. '`colwidths={1=2cm,3=5cm,4=2cm}`' to set the first column to a width of 2 cm, the third to 5 cm and the fourth to 2 cm. (the second column is calculated)

Columns not set this way will be distributed evenly across the remaining horizontal space of the page.

distance Sets the distance between two columns. If omitted set to `2em`.

rulebetween (Flag)[1] Typeset a vertical bar between two columns.

`false` if omitted.

nofirstindent (Flag) Suppress the indentation of the first paragraph in the environment.

`false` if omitted.

sloppy (Flag) Typeset text in the columns in a more sloppy way, preventing overfull hboxes at the cost of larger interword spacing.

`false` if omitted.

sloppyspaces (Flag) Makes spaces in the columns stretchable, preventing overfull hboxes at the cost of even larger interword spacing that `sloppy`.

`false` if omitted.

`\colchunk` Usage: `\colchunk [`⟨*column*⟩`] {`⟨*chunk*⟩`}`.

The macro `\colchunk` sets a chunk of text for the next column. You don't have to set chunks for all columns; the colums not set remain empty. However, the columns are filled from left to right, so if a column inbetween should remain empty, you must say `\colchunk{}` or specify the column to set using the optional parameter (columns start at 1): `\colchunk[2]{...}` sets the text of the second column. Following calls to `\colchunk` without optional parameter fill the third, fourth et cetera column.

After a call to `\colplacechunks`, the column number is reset to one.

`\colplacechunks` Usage: `\colplacechunks`.

The macro `\colplacechunks` places the chunks added with `\colchunk` on the page. If there are no chunks to place, it does nothing.

3 Examples

Two columns, option `rulebetween=true` (which is the same as just saying `rulebetween`):

Erwarten Sie von mir, dass ich rede? – Nein, Mister Bond, ich erwarte von Ihnen, dass Sie sterben!

Erwarten Sie von mir, dass ich rede? – Nein, Mister Bond, ich erwarte von Ihnen, dass Sie sterben!

Do you expect me to talk? – No, Mister Bond, I expect you to die!

Do you expect me to talk? – No, Mister Bond, I expect you to die!

As the german text is slightly longer, let's make the left column alittle bit larger using `colwidths={1=.55\linewidth}`:

Erwarten Sie von mir, dass ich rede? – Nein, Mister Bond, ich erwarte von Ihnen, dass Sie sterben!

Erwarten Sie von mir, dass ich rede? – Nein, Mister Bond, ich erwarte von Ihnen, dass Sie sterben!

Do you expect me to talk? – No, Mister Bond, I expect you to die!

Do you expect me to talk? – No, Mister Bond, I expect you to die!

Three columns, option `nofirstindent=true`:

This is just a short english text, just long enough to fill a few lines.

Dies ist nur ein kurzer deutscher Text, gerade lang genug, um ein paar Zeilen zu fuellen.

This is another short english text, as my french is not that good anymore.

There was an overfull `\hbox` in the previous text. Let's try that again with the option `sloppy`:

This is just a short english text, just long enough to fill a few lines.

Dies ist nur ein kurzer deutscher Text, gerade lang genug, um ein paar Zeilen zu fuellen.

This is another short english text, as my french is not that good anymore.

No overfull hbox this time, but the spacing in the first column is not optimal. You

[1] Meaning that if you just say `rulebetween`, you set the flag, as well as when you say `rulebetween=true`. Saying `rulebetween=false` clears the flag, even though this does not make much sense, as it is the default.

just have to pick what's better. Or you could try the interword spacing provided by the soul package.

Maybe we do not want to fill the first column, but do not want to type `\colchunk{}` either. Then we can say `\colchunk[2]` to directly start with the second column (note that we are using the option `sloppy`, too):

This is just a short english text, just long enough to fill a few lines.

Dies ist nur ein kurzer deutscher Text, gerade lang genug, um ein paar Zeilen zu fuellen.

Note that `parcolumns` does not insert vertical space before or after the environment! In these examples, the space has manually been added with `\addvspace`.

4 Notes

- The columns will always fill the complete width of the page by stretching or shrinking the space between the columns. That means that if the total width of all columns is set to about half the page width, the space between the columns will take up the rest, ignoring whatever was set with the key `distance`.[2]
- Footnotes are not set in columns.
- parcolumns does not work very well with displayed formula. The best way to typeset a `displaymath` et.al. environment is to include it in separate `\colchunk` commands, i.e. (assuming two columns and the same formula in both):

```
... some text placed using \colchunk ...
\colchunk{ % Left column
\begin{displaymath}
x^2 + y^2 = z^2
\end{displaymath}
}
\colchunk{ % Right column
\begin{displaymath}
x^2 + y^2 = z^2
\end{displaymath}
}
\colplacechunks
... some more text placed using \colchunk ...
```

[2]The key `distance` is only used for calculating the width of the columns and is ignored afterwards.

5 Implementation

5.1 Allocation

The current column (starting with one):

```
\newcount\pc@columnctr
```

The total number of columns in the current `parcolums` environment.

```
\newcount\pc@columncount
```

Place a vertical rule between two columns?

```
\newif\ifpc@rulebetween
```

Stores `\everypar` for use in `\parparagraphs`.

```
\newtoks\pc@everypar
```

Note that additional allocations are performed later on-demand in `\pc@alloccolumns`.

5.2 keyvalue keys

`\pc@boolkey` Sets an `if`-condition in `{`⟨*#1*⟩`}` to the value passed as `{`⟨*#2*⟩`}`. If `#2` is `false`, set `if#1` to `false`, else (any other value) to `true`.

Usage: `\pc@boolkey {`⟨*ifcondition*⟩`} {`⟨*value*⟩`}`

```
\def\pc@boolkey#1#2{%
  \edef\@tempa{#2}%
  \edef\@tempb{false}%
  \ifx\@tempa\@tempb%
    \csname #1false\endcsname%
  \else%
    \csname #1true\endcsname%
  \fi%
}
```

Define the keys for the options of `parcolumns`.

```
\define@key{parcolumns}{distance}{%
  \@tempdimc#1\relax%
}
\define@key{parcolumns}{rulebetween}[true]{%
  \pc@boolkey{pc@rulebetween}{#1}%
}
\define@key{parcolumns}{nofirstindent}[true]{%
  \pc@boolkey{@tempswa}{#1}%
```

If the indentation of the first line of the first paragraph should be suppressed, change `\pc@everypar` accordingly. The token register is reset after the placing of the first row of chunks in `\colplacechunks`.

What exactly are we doing? First we assign box 0 the contents of `\lastbox`. At the beginning of a paragraph `\lastbox` contains the glue inserted for the indentation of the first line. By assigning this box to box 0, we remove the indentation. We do this in a group as not to change the contents of box 0.

Afterwards we set `\everypar` to nothing. This is necessary because we only want to suppress the indentation for the first paragraph of the `parcolumns` environment, not every paragraph. We set `\everypar` and not `\pc@everypar` to nothing, because `\pc@everypar` is only a temporary storage that will be assigned to `\everypar` when used.

```
  \if@tempswa\pc@everypar{{\setbox\z@\lastbox}\everypar{}}\fi%
}
\define@key{parcolumns}{sloppy}[true]{%
  \pc@boolkey{@tempswa}{#1}%
```

If sloppy typesetting is asked for, we set `\hbadness` and `\tolerance` to 10000, so that TEX breaks lines whenever possible, even if this means high interword spacing.

```
  \if@tempswa%
    \hbadness\@M%
    \tolerance\@M%
  \fi%
}
\define@key{parcolumns}{sloppyspaces}[true]{%
  \pc@boolkey{@tempswa}{#1}%
```

If sloppy spaces are asked for, we make the space stretchable:

```
  \if@tempswa%
    \spaceskip.3333em\@plus1em %
  \fi%
}
```

We save the key-value-list containing widths of the columns in `\toks@` for later use in `\pc@setcolumnwidths`.

```
\define@key{parcolumns}{colwidths}{%
  \toks@{#1}%
}
```

5.3 Main environments and macros

`parcolumns` Environment for a number of columns of text set in parallel (see section 2 on page 2 for the possible options)

Usage: `\begin{parcolumns}` `[`⟨*options*⟩`]` `{`⟨*number of columns*⟩`}` ... `\end{parcolums}`

```
\newenvironment{parcolumns}[2][]{%
  \pc@rulebetweenfalse%
```

`\if@tempswa` is `true`, if the indentation of the first line should be suppressed, otherwise `false`. Default is `false`:

```
  \@tempswafalse%
```

`\@tempdimc` contains the space between two columns. Default is `2em`:

```
  \@tempdimc2em\relax%
```

We set the options:

```
  \toks@{}%
  \setkeys{parcolumns}{#1}%
```

We store the total number of columns and reset the counter for the current column:

```
  \pc@columncount#2 %
  \pc@columnctr\z@%
```

We allocate the columns and set their widths:

```
  \pc@alloccolumns%
  \pc@setcolumnwidths%
```

We switch to vertical mode:

```
  \endgraf%
```

As we are changing `\everypar`, we need to make sure that the most important flag is reset, which normally happens in the `\everypar` of a `\section`-command: `\if@nobreak`. (otherwise the next section is typeset directly after the text of the previous section instead of leaving some space)[3]

```
  \@nobreakfalse%
```

We reset `\everypar`, as it is of no use for us and can only screw up things badly:

```
  \global\everypar{}%
}{%
```

At the end of the `parcolumns`-environment ... just in case, we place the last chunks, if not already done so:

```
  \colplacechunks%
  \endgraf%
```

We reset `\clubpenalty` globally to its normal value (the `\global` makes sure that if `\everypar` should have reset `\clubpenalty`, it is reset now).

```
  \global\clubpenalty\@clubpenalty%
```

We suppress the indentation of the next paragraph immediately following the environment:

```
  \@doendpe%
}
```

`\colchunk` Sets the text for the next chunk of the next column.

Usage: `\colchunk [`⟨*column*⟩`] {`⟨*chunk*⟩`}`. (note that the `{` and `}` are *not* optional, even if the chunk consists of only one token!)

We need two macros for handling the optional parameter ⟨*column*⟩, `\colchunk` and `\colchunk@`. First we define `\colchunk`:

```
\newcommand{\colchunk}{\@testopt\colchunk@{}}%
```

[3] I discovered this the hard way, spending one or two hours wondering why the spacing between two sections was much too small.

`\colchunk@` What are we doing now? By suffixing the last (optional) parameter by `#`, we tell TEX that the last parameter of `\colchunk` is to be delimited by a right brace `{`, or TEX will complain with a more comprehensible message that without the `#`: 'Use of `\colchunk` doesn't match its definition.' instead of 'Missing `{` inserted.'

```
\long\def\colchunk@[#1]#{%
```

We check if the optional parameter is not empty. Then we use it as the column number, otherwise we simply pick the next column:

```
  \ifx\\#1\\%
    \advance\pc@columnctr\@ne%
  \else%
    \pc@columnctr#1\relax%
  \fi%
```

If we try to add a column past the last one, we display an error message:

```
  \ifnum\pc@columnctr>\pc@columncount%
    \PackageError{parcolumns}{The column \number\pc@columnctr\space%
      is too large}{Only \number\pc@columncount\space columns are%
      \space allowed.}
```

As we cannot simply skip the chunk (we could gobble what follows after the macro, but this would require a lot of jumping, and just for handling a small mistake ...), we simply set the last column:

```
    \pc@columnctr\pc@columncount%
  \fi%
```

We zero the `\clubpenalty` that could have been changed, i.e. by `\everypar` of a `\section`-command: `\section` changes the clubpenalty to prevent a break between the first two lines of the paragraph following immediately after the section; as we split off line after line when typesetting the two columns, this would make splitting of a single line impossible (`\vsplit` uses the same logic as page-breaking), thus resulting in a lot of overfull vboxes and weird spacing inbetween, as *two* lines would be split off.

We do this every time we add chunks just in case some macro in the chunks has changed `\clubpenalty`. Note that we cannot prevent a macro in the text of the chunk to change the `\clubpenalty`.[4] But if it is changed, at least it will not stay changed (though the typeset columns will still look bad).

```
  \clubpenalty\z@%
```

The same goes for the other penalties TEX will insert between two lines, as we absolutely, positively *must* be able to break between any two lines:[5]

```
  \interlinepenalty\z@%
  \displaywidowpenalty\z@%
  \widowpenalty\z@%
  \brokenpenalty\z@%
```

[4] Well we could, by redefining the clubpenalty to be a macro that simply throws its parameter away, and using some tricks to make this macro behave like a register, therefore completely ignoring any change of the clubpenalty.

[5] C.f. chapter 14 of the TEXbook

We set `\everypar` to our self-defined `\pc@everypar` to suppress the indentation of the first paragraph if so desired:

```
\everypar\expandafter{\the\pc@everypar}%
```

After the next assignment, insert the width of the column:

```
\afterassignment\pc@setcolumnwidth%
```

We typeset the chunk's text into the box `\pc@column@`⟨*column counter*⟩. The text of the chunk follows the macro, meaning that the last line looks like this: `\vbox{`⟨*chunk text*⟩`}`.

But what about the width of the box? `\hsize` must be set in the vbox in order to make it the correct width! We achieve this using the `\afterassignment` above: After the assignment of the chunk's text to the box, we are inside the box, therefore the contents of `\pc@setcolumnwidth` is inserted at the very beginning of the vbox, setting the correct width.

Why don't we simply make `\colchunk` take one parameter that contains the text of the chunk? Because in that case, macros that change catcodes like `\verb` would be prohibited, which would restrict this package somewhat. Also it would be slower and would use more memory.

```
  \expandafter\setbox\csname pc@column@\number\pc@columnctr\endcsname%
    \vbox%
}
```

`\colplacechunks` Places all chunks set with `\colchunk`.

```
\newcommand{\colplacechunks}{%
```

If there are any chunks to place:

```
  \ifnum\pc@columnctr>\z@%
```

We place them:

```
    \pc@placeboxes%
```

We reset the column counter:

```
    \pc@columnctr\z@%
```

We clear `\pc@everypar`, because we only want to suppress the indentation of the first paragraph of each column at the very top of the `parcolumns` environment, not of the first paragraph of every call to `\colchunk`.

```
    \pc@everypar{}%
  \fi%
}
```

5.4 Internal environments and macros

`\pc@placeboxes` Places the prepared boxes (containing the chunks) on the page.

```
\def\pc@placeboxes{%
```

We assume we don't have to perform another line (`\@tempa` contains what we have to to after we are finished with this macro). The assignment is global because later on when changing `\@tempa` we are in a group:

```
\global\let\@tempa\relax%
\count@\z@%
```

We create a hbox. Inside a hbox, vboxes are put horizontally next to each other and are aligned on their baseline:

```
\hb@xt@\linewidth{%
```

Before doing any work, we change a few parameters:

We prevent warnings of overfull and underfull vboxes as they can happen, but we do not really care (happens when we `\vsplit` the top off the chunks, this is okay):

```
  \vfuzz30ex %
  \vbadness\@M%
```

We prevent vertical glue to be insert when `vsplit`ting a vbox (otherwise it screws up the spacing).

```
  \splittopskip\z@skip%
```

Now we loop over all the prepared boxes. We can use `\loop` here as we are in a group (begun by `\hbox`). Otherwise we would have to open a group manually or loop manually, as to not screw up a `\loop` outside the macro:

```
  \loop\ifnum\count@<\pc@columncount%
    \advance\count@\@ne%
```

If the box `\pc@column@`⟨*counter*⟩ is empty, we simply insert a hskip the width of the box (saved in `\pc@column@width@`⟨*counter*⟩):

```
    \expandafter\ifvoid\csname pc@column@\number\count@%
      \endcsname%
      \hskip\csname pc@column@width@\number\count@\endcsname%
    \else%
```

Otherwise we `\vsplit` the first line of the box (at the same time removing it from `\pc@column@`⟨*counter*⟩) and save it in `\@tempboxa`. Then, we strip this resulting box of its enclosing vbox and put it into another vbox, which we put into the hbox begun a few lines ago.

Why is this so complicated? To ensure that proper vertical glue is inserted (otherwise the spacing between the lines would be wrong).

```
      \expandafter\setbox\expandafter\@tempboxa\expandafter%
        \vsplit\csname pc@column@\number\count@\endcsname%
          to \dp\strutbox%
      \vbox{\unvbox\@tempboxa}%
    \fi%
```

If the remaining box is not empty, we have to perform at least another line:

```
    \expandafter\ifvoid\csname pc@column@\number\count@%
```

```
        \endcsname\else%
        \global\let\@tempa\pc@placeboxes%
      \fi%
```

If this is not the last column, we put a strut into the hbox to ensure propery vertical spacing:

```
      \ifnum\count@<\pc@columncount%
        \strut%
```

If a vertical line should be placed between two columns, we insert it now, centering it between two `\hfill`s. Otherwise, we simply insert a `\hfill` that stretches as much as possible, pushing the right column to the right margin. (see also section 4 on page 4)

```
        \hfill%
        \ifpc@rulebetween%
          \vrule%
          \hfill%
        \fi%
      \fi%
    \repeat%
  }%
```

If necessary, we perform another line:

```
  \@tempa%
}
```

`\pc@alloccolumns` Allocates a number of `\box`es, named `\pc@column@1`, `\pc@column@2` et cetera, if not already allocated.

Also allocates a number of `\dimen`s, named `\pc@column@width@1` et cetera.

If the `\box`es and `\dimen`s are already allocated, they are cleared (`\box`es) or set to zero (`\dimen`s).

```
\def\pc@alloccolumns{%
  \count@\z@%
  \loop\ifnum\count@<\pc@columncount%
    \advance\count@\@ne%
    \@ifundefined{pc@column@\number\count@}{%
      \expandafter\newbox\csname pc@column@\number\count@%
        \endcsname%
      \expandafter\newdimen\csname pc@column@width@\number%
        \count@\endcsname%
    }{%
      \setbox0\box\csname pc@column@\number\count@\endcsname%
      \csname pc@column@width@\number\count@\endcsname\z@%
    }%
  \repeat%
}
```

`\pc@setcolumnwidths` Sets the widths of all columns. The defined widths have been temporarily stored in `\toks@`. `\@tempdimc` contains the space between two columns.

```
\def\pc@setcolumnwidths{%
  \expandafter\processkeyvalues\expandafter{\the\toks@}%
    \pc@setsinglecolwidth%
```

`\@tempdima` will contain the total width of all columns that have a known width (read: have had their width set via the parameter `colwidths`)

```
  \@tempdima\z@%
```

`\@tempcnta` will contain the count of columns that an unknown width (read: have had their width *not* set via the parameter `colwidths`, thus now having a width of zero points, as the widths of all columns have been reset in `\pc@alloccolumns`):

```
  \@tempcnta\z@%
```

We calculate the total width of all columns known. We count how many columns have an unknown width:

```
  \count@\z@%
  \loop\ifnum\count@<\pc@columncount%
    \advance\count@\@ne%
    \@tempdimb\csname pc@column@width@\number\count@\endcsname%
    \advance\@tempdima\@tempdimb%
    \ifnum\@tempdimb=\z@%
      \advance\@tempcnta\@ne%
    \else%
      \PackageInfo{parcolumns}{Width of column \number\count@%
        \space set to \the\@tempdimb}
    \fi%
  \repeat%
```

If at least one column has an unknown width:

```
  \ifnum\@tempcnta>\z@%
```

`\@tempdimc` contains the distance between columns. We calculate the space left for the columns with an unknown width.

`\@tempdimb` will contain the sum of the space between all the columns ...

```
    \@tempdimb\@tempdimc%
    \multiply\@tempdimb\pc@columncount%
    \advance\@tempdimb-\@tempdimc%
```

... plus the sum of the width of all columns with a known width:

```
    \advance\@tempdimb\@tempdima%
```

`\@tempdima` will contain the space for each column with an unknown width:

```
    \@tempdima\linewidth%
    \advance\@tempdima-\@tempdimb%
    \divide\@tempdima\@tempcnta%
```

We set the widths of the colums with an unknown width:

```
    \count@\z@%
```

```
    \loop\ifnum\count@<\pc@columncount%
      \advance\count@\@ne%
      \ifnum\csname pc@column@width@\number\count@\endcsname=\z@%
        \csname pc@column@width@\number\count@\endcsname\@tempdima%
        \PackageInfo{parcolumns}{Width of column \number\count@%
          \space calculated as \the\@tempdima}
      \fi%
    \repeat%
  \fi%
}
```

`\pc@setsinglecolwidth` Usage: `\pc@setsinglecolwidth {`⟨*column*⟩`} {`⟨*width*⟩`}`.

Sets the width of the column ⟨*column*⟩ to the width ⟨*width*⟩. Displays an error message if the column is not valid.

```
\def\pc@setsinglecolwidth#1#2{%
  \@ifundefined{pc@column@width@\number#1}{
    \PackageError{parcolumns}{`#1' is not a valid column number!}%
      {\@ehc}%
  }{%
    \csname pc@column@width@\number#1\endcsname=#2\relax%
  }%
}
```

`\pc@setcolumnwidth` Sets `\hsize` to the width of a column (stored in `\pc@column@width@`⟨*pc@columnctr*⟩) and enters horizonal mode.

```
\def\pc@setcolumnwidth{%
  \hsize\csname pc@column@width@\number\pc@columnctr\endcsname%
  \linewidth\hsize%
  \leavevmode%
}
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.