The nccboxes package[*]

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2005/02/07

The package implement special boxes and struts from NCC-LaTeX.

1 User Interface

`\jhbox` This macro specifies a horizontal box whose width is calculate using a prototype and alignment position is specified in the same manner as in the `\makebox` command. The syntax:

`\jhbox{`⟨*prototype*⟩`}[`⟨*pos*⟩`]{`⟨*text*⟩`}`

Here ⟨*prototype*⟩ is a text whose width will be the width of generated box, ⟨*pos*⟩ is an alignment parameter (`l`, `c`, `r`, or `s`; default is `c`).

`\jvbox` This macro specifies a horizontal box whose height, depth, and vertical alignment is calculated using a prototype. The syntax:

`\jvbox{`⟨*prototype*⟩`}[`⟨*pos*⟩`]{`⟨*text*⟩`}`

The ⟨*text*⟩ argument is vertically aligned with respect to the strut defined by the ⟨*prototype*⟩ parameter. The optional ⟨*pos*⟩ parameter defines an alignment position (`t`, `c`, or `b`; default is `c`). If `t` is used, the ⟨*text*⟩ is raised in such a way that its height will be equal to the height of the prototype's strut. For the `b` case, the depths will be equal, and, for the `c` case, the ⟨*text*⟩ is vertically centered with respect to the prototype's strut. The height and depth of the prepared box are calculated as a maximum between the corresponding parameters of the ⟨*prototype*⟩ and the vertically adjusted ⟨*text*⟩.

`\jparbox` This macro prepares a paragraph box of the required width and vertically aligns it with respect to the prototype just in the same manner as the `\jvbox`. The syntax:

`\jparbox{`⟨*prototype*⟩`}[`⟨*pos*⟩`]{`⟨*width*⟩`}{`⟨*text*⟩`}`

The ⟨*prototype*⟩ and ⟨*pos*⟩ parameters have the same meaning as described for `\jvbox`. The ⟨*width*⟩ is the width of the paragraph box and the ⟨*text*⟩ is the box content.

[*]This file has version number v1.2, last revised 2005/02/07.

`\addbox` This macro specifies a horizontal box whose height and depth are adjusted using the given values. The syntax:

```
\addbox{⟨height-adjust⟩}{⟨depth-adjust⟩}{⟨text⟩}
```

For example, `\addbox{.5ex}{.5ex}{text}` increases the height and depth of produced box on `0.5ex`.

`\pbox` This macro implements a simple one-column table. The syntax:

```
\pbox[⟨pos⟩]{⟨body⟩}
```

The ⟨*pos*⟩ parameter may consist of two letters defining a relative alignment of the table rows in the column (`l`, `c`, or `r`) and the vertical alignment of the whole table with respect to surrounding text (`t`, `c`, or `b`). Centering is the default alignment. The distance between table rows does not depend on the `\arraystretch` value.

`\picbox` The `\picbox{`⟨*body*⟩`}` macro is equivalent to

```
\begin{picture}(0,0)(0,0)⟨body⟩\end{picture}.
```

To prepare fancy tables, the following commands can be used:

`\Strut` `\Strutletter` The `\Strut/`⟨*value*⟩`/` command is a special strut whose height and depth are calculated from the strut prototype command `\Strutletter` (letter `A` by default) as follows: if ⟨*value*⟩ is positive, the full height of the current `\strutbox` multiplied by the ⟨*value*⟩ is added to the height of strut prototype, otherwise the depth of strut prototype increases with the modulus of ⟨*value*⟩ multiplied by the full height of `\strutbox`. For example, `\Strut/1/` inserts a strut which height exceeds the height of the letter `A` from the current font on the interline distance. A natural length is also possible as a value of `\Strut`'s parameter. So, the `\Strut/2mm/` means a strut with the height exceeding the height of strut letter over 2 mm. The `\Strut` without parameter is equal to `\Strut/0/`. Spaces after the `\Strut` are ignored.

`\tstrut` `\bstrut` `\tbstrut` `\Strutstretch` The `\tstrut`, `\bstrut`, and `\tbstrut` commands insert struts exceeding the height, depth, and both height and depth of the strut prototype `\Strutletter` by a special small amount. This amount is calculated in such a way that the full height of `\tbstrut` will be equal to 1.5 of full height of the current `\strutbox`. The stretch factor 1.5 is specified in the `\Strutstretch` command. These commands are used in tables to insert a space between a horizontal line and a table row. But if the height and depth of row contents exceeds the height and depth of inserted strut, the inserted strut will take no effect.

`\cbox` The `\cbox/`⟨*value*⟩`/[`⟨*pos*⟩`]{`⟨*body*⟩`}` command prepares a box whose body is a one-column table. Its height and depth are enlarged using `\tstrut` at the beginning and `\bstrut` at the end of body. The horizontal alignment (`l`, `c`, or `r`) in the column and the vertical alignment (`t`, `c`, or `b`) are defined in the ⟨*pos*⟩ parameter. Centered alignment is used by default. The resulting box is vertically aligned with respect to the `\Strut/`⟨*value*⟩`/` using the `\jvbox` command. The `\cbox*` command does the same but vanishes the height and depth of the resulting box. The `\cbox` command is used in the headers of tables. Its star form is useful in cells having vertical spans.

`\cboxstyle` The `\cboxstyle` specifies a style applied to all `\cbox` commands. It can set a font size, shape, color, etc. The default value of `\cboxstyle` is empty.

We demonstrate the usage of struts and `\cbox` on the following example:

Vertically spanned head	**Simple head**	**Very long head of two lines**	
	subhead	**subhead**	**subhead**
Text	field	field	field
Text	field	field	field
Text	field	field	field

It was produced as follows:

```
\begin{center}
  \renewcommand\cboxstyle{\small\bf}
  \setlength{\tabcolsep}{10pt}
  \begin{tabular}{|l|c|c|c|}\hline
    \cbox*/-1.5/{Vertically\\spanned head} & \cbox{Simple head}
    &\multicolumn2{c|}{\cbox{Very long head\\of two lines}}\\
    \cline{2-4}
    &\cbox{subhead} &\cbox{subhead} &\cbox{subhead}\\\hline
    \Strut/1/ Text & field & field & field \\
              Text & field & field & field \\
    \bstrut   Text & field & field & field \\\hline
  \end{tabular}
\end{center}
```

`\tc` To center a table field, the `\tc{`⟨*field*⟩`}` command is introduced since version 1.2 of the package. It inserts `\hspace*{\fill}` before and after the ⟨*field*⟩.

2 The Implementation

`\addbox` The implementation of `\addbox{`⟨*height-adjust*⟩`}{`⟨*depth-adjust*⟩`}{`⟨*text*⟩`}`. We use the `\setlength` in calculations of box's height and depth for compatibility with the `calc` package.

```
⟨*package⟩
\newcommand*{\addbox}[3]{%
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%
    \advance\@tempdima \ht\@tempboxa
    \ht\@tempboxa \@tempdima
    \setlength\@tempdima{#2}%
    \advance\@tempdima \dp\@tempboxa
    \dp\@tempboxa \@tempdima
    \leavevmode\box\@tempboxa
  \@end@tempboxa
}
```

`\jhbox` The implementation of `\jhbox{`⟨*prototype*⟩`}[`⟨*pos*⟩`]{`⟨*text*⟩`}` is very simple:

```
\newcommand*{\jhbox}[1]{\settowidth\@tempdima{#1}\makebox[\@tempdima]}
```

`\jvbox` The `\jvbox{`⟨*prototype*⟩`}[`⟨*pos*⟩`]{`⟨*text*⟩`}` is implemented as follows. We prepare a vertical strut in zero box using the ⟨*prototype*⟩. Then we vertically adjust the content of the `\jvbox` and put the strut and the adjusted box.

```
\newcommand*{\jvbox}[1]{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}%
  \setbox\z@\hbox{\vrule \@width\z@ \@height\ht\z@ \@depth\dp\z@}%
  \NCC@jvbox
}
\newcommand*{\NCC@jvbox}[2][]{%
  \setbox\@tempboxa\hbox{\color@begingroup#2\color@endgroup}%
  \let\m@t\vss \let\m@b\vss
  \@tfor\@tempa :=#1\do {%
    \expandafter\let\csname m@\@tempa\endcsname\relax}%
  \@tempdima\ht\z@ \advance\@tempdima -\ht\@tempboxa
  \ifx\m@t\relax \else
    \@tempdimb\dp\@tempboxa \advance\@tempdimb -\dp\z@
    \ifx\m@b\relax \@tempdima \@tempdimb \else
      \advance\@tempdimb \@tempdima \@tempdima .5\@tempdimb
    \fi
  \fi
  \leavevmode \box\z@ \raise\@tempdima\box\@tempboxa
}
```

`\jparbox` The implementation of `\jparbox{`⟨*prototype*⟩`}[`⟨*pos*⟩`]{`⟨*width*⟩`}{`⟨*body*⟩`}` is based on `\jvbox`, but we prepare the ⟨*body*⟩ in the vertical box.

```
\newcommand*{\jparbox}[1]{%
  \@ifnextchar[{\NCC@jparbox{#1}}{\NCC@jparbox{#1}[]}%
}
\long\def\NCC@jparbox#1[#2]#3#4{%
  \@begin@tempboxa\vtop{\setlength\@tempdima{#3}%
    \hsize\@tempdima\@parboxrestore#4\@@par}%
    \setlength\@tempdima{#3}% vbox containing only display equations can
    \wd\@tempboxa\@tempdima % have lesser width. We correct it here
    \jvbox{#1}[#2]{\box\@tempboxa}%
  \@end@tempboxa
}
```

`\pbox` Now we implement the `\pbox[`⟨*pos*⟩`]{`⟨*body*⟩`}` command. It is a simple one-column table. The `\arraystretch` has no effect on it. The ⟨*pos*⟩ is a combination of vertical (tbc) and horizontal (lcr) positions. For example, `lt` means left adjusted table with first line on the base line.

```
\newcommand*{\pbox}[2][]{%
  \let\m@l\hss \let\m@r\hss \let\m@t\vss \let\m@b\vss
  \@tfor\@tempa:=#1\do{%
```

```
    \expandafter\let\csname m@\@tempa\endcsname\relax%
  }%
  \leavevmode\hbox{\color@begingroup
    $\ifx\m@t\relax \vtop  \else\ifx\m@b\relax \vbox\else \vcenter\fi\fi
    \bgroup \baselineskip\z@\lineskip\z@
      \def\\{\strut\@stackcr}%
      \halign{\m@l\ignorespaces ##\unskip\m@r\cr #2\strut\crcr}%
    \egroup$\color@endgroup
  }%
}
```

`\picbox` The `\picbox{`⟨*body*⟩`}` command:

```
\newcommand*{\picbox}[1]{%
  \setbox\@tempboxa\hb@xt@\z@{\ignorespaces#1\hss}%
  \ht\@tempboxa\z@\dp\@tempboxa\z@
  \leavevmode\box\@tempboxa
}
```

`\Strutletter` `\Strutstretch` Here we specify macros for preparing special struts. The `\Strutletter` is the prototype for special struts. The `\Strutstretch` is a stretch of line height in `\cbox` with respect to `\strut`. We prepare special struts in the `\NCC@strutbox`. The `\NCC@strutsep` is a half of difference between stretched `\strut` and the full height of the `\Strutletter`.

```
\newcommand{\Strutletter}{A}
\newcommand{\Strutstretch}{1.5}
\newsavebox\NCC@strutbox
\newdimen\NCC@strutsep
```

`\NCC@setstrut` The `\NCC@setstrut{`⟨*command*⟩`}/`⟨*value*⟩`/` tests the sequence `/`⟨*value*⟩`/`, prepares the specified strut in the `\NCC@strutbox`, calculates the `\NCC@strutsep`, and then calls the ⟨*command*⟩. The `/`⟨*value*⟩`/` sequence is optional. If it is omitted, `/0/` is supposed.

```
\def\NCC@setstrut#1{%
  \setbox\NCC@strutbox\hbox{\vphantom{\Strutletter}}%
  \@tempdima\ht\strutbox \advance\@tempdima\dp\strutbox
  \NCC@strutsep \Strutstretch\@tempdima
  \advance\NCC@strutsep -\ht\NCC@strutbox
  \advance\NCC@strutsep -\dp\NCC@strutbox
  \NCC@strutsep .5\NCC@strutsep
  \@ifnextchar/{\NCC@setstrutn{#1}}{\NCC@setstrutl{#1}\z@}%
}
\def\NCC@setstrutn#1/#2/{\NCC@setstrutl{#1}{#2\@tempdima}}
\def\NCC@setstrutl#1#2{%
  \@defaultunits\@tempdima#2\relax\@nnil
  \ifdim\@tempdima>\z@
    \advance\@tempdima \ht\NCC@strutbox
    \ht\NCC@strutbox \@tempdima
  \else
    \@tempdima -\@tempdima
    \advance\@tempdima \dp\NCC@strutbox
    \dp\NCC@strutbox \@tempdima
  \fi
  #1%
}
```

`\Strut` Now we define the `\Strut/`⟨*value*⟩`/`. It is quite simple:

```
\newcommand{\Strut}{%
  \NCC@setstrut{\leavevmode\copy\NCC@strutbox\ignorespaces}%
}
```

`\tstrut` `\bstrut` `\tbstrut` Next we define `\tstrut`, `\bstrut`, and `\tbstrut` via the `\addbox` command. All these struts use the `\NCC@setstrut` to calculate special strut parameters.

```
\newcommand{\tstrut}{%
  \NCC@setstrut{}\addbox\NCC@strutsep\z@{\copy\NCC@strutbox}%
}
\newcommand{\bstrut}{%
  \NCC@setstrut{}\addbox\z@\NCC@strutsep{\copy\NCC@strutbox}%
}
\newcommand{\tbstrut}{%
  \NCC@setstrut{}\addbox\NCC@strutsep\NCC@strutsep{\copy\NCC@strutbox}%
}
```

`\cbox` `\cboxstyle` Now, we define the `\cbox/`⟨*value*⟩`/[`⟨*pos*⟩`]{`⟨*body*⟩`}` command and its star-form. It is useful in headers of tables. The `\cboxstyle` is a styling command applied to every `\cbox`.

```
\newcommand{\cboxstyle}{}
\newcommand{\cbox}{%
  \@ifstar{\def\@tempa{\ht\@tempboxa\z@ \dp\@tempboxa\z@}\NCC@xcbox}%
          {\let\@tempa\relax\NCC@xcbox}%
}
\def\NCC@xcbox{%
  \leavevmode \hbox\bgroup\color@begingroup
  \cboxstyle\NCC@setstrut{\NCC@ycbox}%
}
\newcommand*{\NCC@ycbox}[2][]{%
  \setbox\@tempboxa\hbox{%
    \jvbox{\addbox\NCC@strutsep\NCC@strutsep{\copy\NCC@strutbox}}[#1]%
      {\pbox[#1t]{\tstrut\ignorespaces #2\unskip\bstrut}}}%
  \@tempa \box\@tempboxa
  \color@endgroup\egroup
}
```

`\tc` And finally, we define the `\tc{`⟨*field*⟩`}` command.

```
\newcommand{\tc}[1]{\hspace*{\fill}#1\hspace*{\fill}}
⟨/package⟩
```