The \texttt{eqparbox} package*

Scott Pakin
\texttt{scott+eqp@pakin.org}

2004/08/02

Abstract

The \texttt{eqparbox} package makes it easy to define a group of \texttt{parbox}es whose members all have the same width, the natural width of the widest member. A document can contain any number of groups, and each group can contain any number of members. This simple, equal-width mechanism can be used for a variety of alignment purposes, as is evidenced by the examples in this document.

1 Motivation

Let’s start with a little test. How would you typeset Table 1, in which the numbers are right-justified relative to each other but centered as a group within each column. And second, how would you typeset the résumé excerpt shown in Figure 1 while meeting the following requirements:

1. The header columns must be left-justified relative to each other.
2. The headers columns should be evenly spaced across the page.
3. Page breaks should be allowed within the résumé.

The two questions can be answered the same way: by putting various blocks of text into equal-widthed boxes. If the data in Table 1 are put into equal-sized \texttt{parbox}es, each containing a \texttt{raggedleft} for right-justification, the \texttt{parbox}es can then be centered to achieve the desired result. Similarly, if the company names in Figure 1 are both put in a \texttt{parbox} as wide as “Thingamabobs, Ltd.,” the job titles in a \texttt{parbox} as wide as “Senior Widget Designer,” and the dates in a \texttt{parbox} as wide as “1/95–present,” then they can be spaced evenly by separating them with \texttt{hfill}s.

The problem is in choosing the width for each set of \texttt{parbox}es. For Table 1 this isn’t too difficult, because digits are the same width as each other in most fonts. Each \texttt{parbox}, therefore, need be only as wide as the largest sequence of

\footnote{This file has version number v2.1, last revised 2004/08/02.}
Table 1: Sample sales data

<table>
<thead>
<tr>
<th>Product</th>
<th>Sales (in millions)</th>
<th>October</th>
<th>November</th>
<th>December</th>
</tr>
</thead>
<tbody>
<tr>
<td>Widgets</td>
<td>55.2</td>
<td>89.2</td>
<td>57.9</td>
<td></td>
</tr>
<tr>
<td>Doohickeys</td>
<td>65.0</td>
<td>64.1</td>
<td>9.3</td>
<td></td>
</tr>
<tr>
<td>Thingamabobs</td>
<td>10.4</td>
<td>8.0</td>
<td>109.7</td>
<td></td>
</tr>
</tbody>
</table>

Widgets, Inc. **Senior Widget Designer** 1/95–present

- Supervised the development of the new orange and blue widget lines.
- Improved the design of various widgets, making them less sticky and far less likely to explode.
- Made widget management ten times more cost-effective.

Thingamabobs, Ltd. **Lead Engineer** 9/92–12/94

- Found a way to make thingamabobs run on solar power.
- Drafted a blueprint for a new doohickey-compatibility module for all cool-mint thingamabobs.
- Upgraded superthingamabob specification document from Microsoft Word to $\LaTeX$.

Figure 1: Excerpt from a sample résumé
digits expected. Figure 1 is more of a bother. The user must typeset the résumé
once to see which entry in each column is the widest and then assign lengths
appropriately:

\newlength{\placewidth}
\settowidth{\placewidth}{Thingamabobs, Ltd.} \% Employment 2
\newlength{\jobtitlewidth}
\settowidth{\jobtitlewidth}{Senior Widget Designer} \% Employment 1
\newlength{\dateswidth}
\settowidth{\dateswidth}{1/95--present} \% Employment 1

Every time a piece of information changes, it must be changed in two places: in the
résumé itself and in the \settowidth command. When employment information
is added or deleted, the \settowidth commands must be modified to reflect the
new maximum-widthed entry in each column. If only there were a simpler way to
keep a set of \parboxes as wide as the widest entry in the set . . .

That simpler way is the eqparbox package. eqparbox exports a macro,
\eqparbox, which works just like \parbox, except that instead of specifying the
width of the box, one specifies the group that the box belongs to. All boxes in
the same group will be typeset as wide as the widest member of the group. In
that sense, an \eqparbox behaves like a cell in an 1, c, or r column in a tabular;
\eqparboxes in the same group are analogous to cells in the same column.

2 Usage

\eqparbox The primary macro in the eqparbox package is \eqparbox. Usage is almost iden-
tical to that of \parbox:

\eqparbox [⟨pos⟩] [⟨height⟩] [⟨inner-pos⟩] {⟨tag⟩} {⟨text⟩}

The only difference is that, where \parbox has its ⟨width⟩ argument, \eqparbox
has ⟨tag⟩. (For a description of the remaining arguments, look up \parbox in any
LATEX 2ε book or in the usrguide.tex file that comes with LATEX 2ε.) ⟨tag⟩ can
be any valid identifier. All \eqparboxes with the same tag will be typeset in a
box wide enough to hold the widest of them. Discounting \TeX ’s limitations, any
number of tags can be used in the same document, and any number of \eqparboxes
can share a tag.

The only catch is that latex will need to be run a second time for the various
box widths to stabilize.

\eqboxwidth It is sometimes useful to take the width of an \eqparbox to use in other \LaTeX
commands. While the width can be determined by creating an \eqparbox and
using \settowidth to measure it, the eqparbox package defines a convenience
routine called \eqboxwidth that achieves the same result.

\eqboxwidth makes it easy to typeset something like Table 2. Table 2’s only
column expands to fit the widest cell in the column, excluding the final cell. The
Table 2: A \texttt{tabular} that stretches to fit some cells while forcing others to wrap

<table>
<thead>
<tr>
<th>Wide</th>
<th>Wider</th>
<th>Wider than that</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a fairly wide cell</td>
<td></td>
<td></td>
</tr>
<tr>
<td>While this cell’s text wraps, the previous cells (whose text doesn’t wrap) determine the width of the column.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

final cell’s text word-wraps within whatever space is allocated to it. In a sense, the first four cells behave as if they were typeset in an \texttt{l} column, while the final cell behaves as if it were typeset in a \texttt{p} column. In actuality, the column is an \texttt{l} column; an \texttt{eqparbox} for the first four cells ensures the column stretches appropriately, while a \texttt{parbox} of width \texttt{eqboxwidth\{tag\}} in the final cell ensures that the final cell word-wraps.

3 Examples

Figure 1’s headings were typeset with the following code:

\begin{verbatim}
\noindent% 
\eqparbox{place}{\textbf{Widgets, Inc.}} \hfill 
\eqparbox{title}{\textbf{Senior Widget Designer}} \hfill 
\eqparbox{dates}{\textbf{1/95--present}}

\vdots

\noindent% 
\eqparbox{place}{\textbf{Thingamabobs, Ltd.}} \hfill 
\eqparbox{title}{\textbf{Lead Engineer}} \hfill 
\eqparbox{dates}{\textbf{9/92--12/94}}

\vdots
\end{verbatim}

Table 1 was entered as follows:

\begin{verbatim}
\begin{tabular}{@{}lccc@{}} 
\hline 
\multicolumn{3}{c}{Sales (in millions)} \\ \cline{2-4}
\multicolumn{1}{c}{\raisebox{1ex}[2ex]{Product}} & October & November & December \\ \hline
4
\end{tabular}
\end{verbatim}
<table>
<thead>
<tr>
<th>Widgets</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>55.2</td>
<td>89.2</td>
<td>57.9</td>
</tr>
<tr>
<td>Doohickeys</td>
<td>65.0</td>
<td>64.1</td>
<td>9.3</td>
</tr>
<tr>
<td>Thingamabobs</td>
<td>10.4</td>
<td>8.0</td>
<td>109.7</td>
</tr>
</tbody>
</table>

Note that the above can be simplified by defining a macro that combines \eqparbox and \raggedleft. Furthermore, because the numeric data being typeset are all approximately the same width, a single tag could reasonably replace oct, nov, and dec. As it stands, the code serves more as an illustration than as an optimal way to typeset Table 1.

Finally, Table 2 was typeset using the following code:

\begin{tabular}{|@{}l@{}}
\hline
\eqparbox[b]{wtab}{Wide} \\
\eqparbox[b]{wtab}{Wider} \\
\eqparbox[b]{wtab}{Wider than that} \\
\eqparbox[b]{wtab}{This is a fairly wide cell} \\
\parbox[b]{\eqboxwidth{wtab}}{\strut While this cell’s text wraps, the previous cells (whose text doesn’t wrap) determine the width of the column.} \\
\hline
\end{tabular}

As an additional example, consider the paragraphs depicted in Figure 2. We’d like the paragraph labels set on the left, as shown, but we’d also like to allow both intra- and inter-paragraph page breaks. Of course, if the labels are made wider or narrower, we’d like the paragraph widths to adjust automatically. (Can any word processor do that, incidentally?) By using a custom list environment which typesets its labels with \eqparbox, this is fairly straightforward:

\begin{list}{}{%\renewcommand{\makelabel}[1]{\eqparbox[b]{listlab}{#1}}\setlength{\labelwidth}{\eqboxwidth{listlab}}\setlength{\labelsep}{2em}\setlength{\parsep}{2ex plus 2pt minus 1pt}\setlength{\itemsep}{0pt}\setlength{\leftmargin}{\labelwidth+\labelsep}\setlength{\rightmargin}{0pt}}\item[Stuff about me] I am great. Blah, blah, blah, ...


---

Figure 2: Paragraphs with hanging indentation

\item[More stuff] I am wonderful. Blah, blah, blah, ...

\item[The final exciting thing] I am fantastic. Blah, blah, blah, ...

\end{list}

4 Limitations

Unfortunately, \eqparbox has a number of limitations not exhibited by the original \parbox command. First, \eqparbox internally typesets the given text within a tabular environment—specifically, using “@{}l@{}” as the template—in order to determine the text’s natural width. Consequently, commands not valid within such a tabular (e.g., list environments) are also not valid within the (text) argument of \eqparbox. As a corollary, an \eqparbox can appear only where a tabular is also acceptable.

A second limitation is that \eqparbox typesets its (text) argument twice: once within a tabular to determine the natural width and again within a parbox large enough to hold all text associated with tag (tag). This approach may cause unexpected results if (text) is non-idempotent (i.e., has side effects). For example, if (text) increments a counter, the counter will be incremented twice per invocation of \eqparbox.
5 Implementation

The one-sentence summary of the implementation is, “As \texttt{eqparbox} goes along, it keeps track of the maximum width of each box type, and when it’s finished, it writes those widths to the .aux file for use on subsequent runs.” If you’re satisfied with that summary, then read no further. Otherwise, get ready to tackle the following annotated code listing.

\begin{verbatim}
\newif\ifeqp@must@rerun
\eqp@must@rerunfalse
\AtEndDocument{\begingroup
\def\@elt#1{\
\eqp@tempdima\csname eqp@this@#1\endcsname\relax
\eqp@tempdimb\csname eqp@next@#1\endcsname\relax
\ifdim\eqp@tempdima=\eqp@tempdimb
\else
\@latex@warning@no@line{Rerun to correct the width of eqparbox ‘#1’}\
\fi
\immediate\write\@auxout{\string\expandafter\string\gdef\string\csname eqp@this@#1\string\endcsname{\csname eqp@next@#1\endcsname}\
\endgraf
\string\expandafter\string\gdef\string\csname eqp@next@#1\string\endcsname{0pt}\
\endgraf}
\endgraf}
\endgroup}
\def\@elt{
\end{verbatim}

\texttt{\@elt} The \texttt{\@elt} list is of the form “\texttt{\@elt {\texttt{tag}_1}} \texttt{\@elt {\texttt{tag}_2}} …”. We therefore locally define \texttt{\@elt} to take the name of a tag and perform all of the checking described above and then merely execute \texttt{\@elt}. 

\begin{verbatim}
\def\@elt#1{\
\eqp@tempdima\csname eqp@this@#1\endcsname\relax
\eqp@tempdimb\csname eqp@next@#1\endcsname\relax
\ifdim\eqp@tempdima=\eqp@tempdimb
\else
\@latex@warning@no@line{Rerun to correct the width of eqparbox ‘#1’}\
\fi
\immediate\write\@auxout{\string\expandafter\string\gdef\string\csname eqp@this@#1\string\endcsname{\csname eqp@next@#1\endcsname}\
\endgraf
\string\expandafter\string\gdef\string\csname eqp@next@#1\string\endcsname{0pt}\
\endgraf}
\endverbatim
We output the generic “rerun \texttt{latex}” message if we encountered a tag that was not present on the previous run. (This is always the case on the first run or the first run after deleting the corresponding \texttt{.aux} file.

\if\eqp@must@rerun
\@latex@warning@no@line{Rerun to correct eqparbox widths}
\fi

\eqp@storefont
\eqp@restorefont
To find the natural width of a piece of text, we put it in a table and take the width of that. The problem is that font changes are not preserved across line breaks (table cells). We therefore define an \texttt{\eqp@storefont} macro which itself defines an \texttt{\eqp@restorefont} macro that restores the current font and font size to its current state.

\newcommand*{\eqp@storefont}{\%\xdef\eqp@restorefont{\noexpand\usefont{\f@encoding}{\f@family}{\f@series}{\f@shape}\%\noexpand\fontsize{\f@size}{\f@baselineskip}\%\noexpand\selectfont}\%}

The following macro (\texttt{\eqp@settowidth}) requires the \texttt{array} package’s ability to inject code into every cell.

\RequirePackage{array}
\eqp@storefont
\eqp@restorefont
\eqp@settowidth
This macro is just like \texttt{\settowidth}, but it puts its argument in a \texttt{tabular}, which means that it can contain \texttt{\textbackslash}. We use the \texttt{array} package’s \texttt{\textgreater{}} and \texttt{\textless{}} template parameters to inject an \texttt{\eqp@restorefont} at the start of every cell and an \texttt{\eqp@storefont} at the end of every cell. Doing so preserves fonts and font sizes across \texttt{\textbackslash} boundaries, just like \texttt{\parbox}.

\newcommand{\eqp@settowidth}{[2]{\%\settowidth{#1}{\%\eqp@storefont\begin{tabular}{@{}>{\eqp@restorefont}l<\eqp@storefont}@{}}\%\end{tabular}\%\eqp@storefont\textbackslash\{}\%\textbackslash\}}%

\eqparbox
\eqparbox@i
\eqparbox@ii
We want \texttt{\eqparbox} to take the same arguments as \texttt{\parbox}, with the same default values for the optional arguments. The only difference in argument processing is that \texttt{\eqparbox} has a \texttt{\langle tag\rangle} argument where \texttt{\parbox} has \texttt{\langle width\rangle}.

Because \texttt{\eqparbox} has more than one optional argument, we can’t use a single function defined by \texttt{\DeclareRobustCommand}. Instead, we have to split \texttt{\eqparbox} into the following four macros:
\eqparbox Takes zero or more optional arguments. First optional argument defaults to $c$. Calls \eqparbox@i.

\eqparbox@i Takes one or more optional arguments. Second optional argument defaults to \relax. Calls \eqparbox@ii.

\eqparbox@ii Takes two or more optional arguments. Third optional argument defaults to $s$ if either of the first two arguments is absent or to the first argument if both are present. Calls \eqparbox@iii.

\eqparbox@iii Takes three optional arguments and two mandatory arguments. Does all the work for \eqparbox.

Note the direct correspondence between these macros and \texttt{ltboxes.dtx}'s \parbox, \@iparbox, \@iiparbox, and \@iiiparbox macros.

\begin{verbatim}
\DeclareRobustCommand\eqparbox{%
  \@ifnextchar[%
  {\eqparbox@i}%
  {\eqparbox@iii[c][\relax][s]}%
 }
\def\eqparbox@i[#1]{%
  \@ifnextchar[%
  {\eqparbox@ii[#1]}%
  {\eqparbox@iii[#1][\relax][s]}%
 }
\def\eqparbox@ii[#1][#2]{%
  \@ifnextchar[%
  {\eqparbox@iii[#1][#2]}%
  {\eqparbox@iii[#1][#2][#1]}%
 }
\end{verbatim}

\eqparbox@iii The following function does all the real work for \eqparbox. It takes five parameters—⟨pos⟩, ⟨height⟩, ⟨inner-pos⟩, ⟨tag⟩, and ⟨text⟩—and ensures that all boxes with the same tag will be as wide as the widest box with that tag.

To keep track of box widths, \eqparbox makes use of two global variables for each tag: \texttt{\eqp@this@⟨tag⟩} and \texttt{\eqp@next@⟨tag⟩}. \texttt{\eqp@this@⟨tag⟩} is the maximum width ever seen for tag ⟨tag⟩, including in previous \texttt{latex} runs. \texttt{\eqp@next@⟨tag⟩} works the same way but is always initialized to 0.0pt. It represents the maximum width to assume in subsequent \texttt{latex} runs. It is needed to detect whether the dest text with tag ⟨tag⟩ has been removed/shrunken. At the end of a run, \eqparbox prepares the next run (via the \texttt{.aux} file) to initialize \texttt{\eqp@this@⟨tag⟩} to the final value of \texttt{\eqp@next@⟨tag⟩}.

\begin{verbatim}
\def\eqparbox@iii[#1][#2][#3][#4][#5]{%
  \expandafter
  \ifx\csname eqp@this@#4\endcsname\relax
  \else
    \textbf{If we get here, then we've never encountered tag \langle#4\rangle, even in a previous latex run. We request that the user re-run latex This is not always necessary (e.g.,}
  \fi
}
\end{verbatim}
when all uses of the \texttt{\eqparbox} with tag \langle \texttt{tag} \rangle are left-justified), but it’s better to be safe than sorry.

\begin{verbatim}
\global\eqp@must@reruntrue
\expandafter\xdef\csname eqp@this@#4\endcsname{\the\eqp@tempdimb}%
\expandafter\xdef\csname eqp@next@#4\endcsname{\the\eqp@tempdimb}%
\else
If we get here, then we have previously seen tag \langle \texttt{tag} \rangle. We just have to keep track of the maximum text width associated with it.
\eqp@tempdima=\csname eqp@this@#4\endcsname\relax
\ifdim\eqp@tempdima<\eqp@tempdimb
\expandafter\xdef\csname eqp@this@#4\endcsname{\the\eqp@tempdimb}%
\global\eqp@must@reruntrue
\fi
\eqp@tempdima=\csname eqp@next@#4\endcsname\relax
\ifdim\eqp@tempdima<\eqp@tempdimb
\expandafter\xdef\csname eqp@next@#4\endcsname{\the\eqp@tempdimb}%
\fi
\fi
\end{verbatim}

The first time we encounter tag \langle \texttt{tag} \rangle in the current document we ensure \LaTeX{} will notify the user if he needs to re-run \texttt{latex} on account of that tag.

\begin{verbatim}
\@ifundefined{eqp@seen@#4}{\expandafter\gdef\csname eqp@seen@#4\endcsname{}%}{%\@cons\eqp@taglist{{#4}}%}
\end{verbatim}

Finally, we can call \texttt{\parbox}. We pass it \langle \texttt{pos} \rangle, \langle \texttt{height} \rangle, \langle \texttt{inner-pos} \rangle, and \langle \texttt{text} \rangle directly, and we pass it \texttt{\eqp@this@\langle \texttt{tag} \rangle} for its \langle \texttt{width} \rangle argument.

\begin{verbatim}
\eqp@tempdima=\csname eqp@this@#4\endcsname\relax
\ifdim\eqp@tempdima=\csname eqp@this@#4\endcsname\relax
\expandafter\xdef\csname eqp@seen@#4\endcsname{\the\eqp@tempdimb}%
\parbox[#1][#2][#3]{\eqp@tempdima}{#5}%
\end{verbatim}

\texttt{\eqboxwidth} For the times that the user wants to make something other than a box match an \texttt{\eqparbox}’s width, we provide \texttt{\eqboxwidth}. \texttt{\eqboxwidth} returns the width of a box corresponding to a given tag. More precisely, if \texttt{\eqp@this@\langle \texttt{tag} \rangle} is defined, it’s returned. Otherwise, 0pt is returned.

\begin{verbatim}
\newcommand*{\eqboxwidth}[1]{%\@ifundefined{eqp@this@#1}{0pt}{\csname eqp@this@#1\endcsname}}%
\end{verbatim}

\textbf{Per-tag memory usage} The \texttt{eqparbox} package defines three macros for each unique tag: \texttt{\eqp@this@\langle \texttt{tag} \rangle}, \texttt{\eqp@next@\langle \texttt{tag} \rangle}, and \texttt{\eqp@seen@\langle \texttt{tag} \rangle}. Consequently, each unique tag subtracts three strings from \TeX{}’s string pool and three multiletter control sequences from that pool. In addition, each unique tag \langle \texttt{tag} \rangle subtracts |\texttt{\eqp@this@\langle \texttt{tag} \rangle}| + |\texttt{\eqp@next@\langle \texttt{tag} \rangle}| + |\texttt{\eqp@seen@\langle \texttt{tag} \rangle}| = 27 + 3 \times |\langle \texttt{tag} \rangle| string characters from \TeX{}’s pool of string characters. For example, a document that invokes \texttt{\eqparbox} with tags “hello” and “goodbye”
will utilize $3 \times 2 = 6$ strings, $3 \times 2 = 6$ multiletter control sequences, and 
$(27 + 3 \times 5) + (27 + 3 \times 7) = 90$ \TeX{} string characters.

89 \langle \text{package} \rangle

6 Future Work

The following are some of the features people have requested I implement in \texttt{eqparbox}:

- Evaluate \texttt{\eqparbox}'s \langle \text{text} \rangle argument exactly once in case it contains side effects. Currently, \langle \text{text} \rangle is evaluated twice: once to determine its natural size and once to put it in a parbox of the width associated with the given tag. (Feature requested by Bernd Schandl.)

- Get \texttt{eqparbox} to work properly within an \texttt{algor}ithmic environment. (Feature requested by Mike Shell.)

- Support pre-1999 \TeX{} 2\$\zeta$. (Feature requested by Mike Shell.)

One idea I've been toying with that may resolve the first two items in that list is to typeset \langle \text{text} \rangle within a parbox, then check \texttt{\badness} to see if the box was too small for \langle \text{text} \rangle. If not (i.e., the box was not overfull), then the initial parbox can be reused directly without needing to re-typeset \langle \text{text} \rangle. With this approach, \texttt{\eqparbox} will work anywhere \texttt{\parbox} works. Unfortunately, I don’t know how to get at the \texttt{\badness} of the \texttt{\hbox}es that comprise the \texttt{\vbox} utilized by \texttt{\parbox}. If anyone has a suggestion, I’m all ears.

Change History

v1.0
General: Initial version \ldots \ldots 1

v2.0
\texttt{\@elt}: Modified to allow numbers in tag names (suggested by Martin Vaeth) \ldots \ldots \ldots \ldots 7

General: Rewrote to use only two \langle \text{dimen} \rangle s total and the rest macros (problem reported by Gilles Pérez-Lambert and Plamen Tanovski; solution suggested by David Kastrup and Donald Arseneau) \ldots \ldots \ldots \ldots 1

\texttt{\eqp@settowidth}: Modified to store and restore the font across \texttt{\parbox}'s boundaries (suggested by Mike Shell) \ldots \ldots \ldots \ldots 8

\texttt{\eqparbox@iii}: Removed extraneous \texttt{\globals} (suggested by David Kastrup) \ldots \ldots \ldots \ldots 9

v2.1
\texttt{\eqboxwidth}: Rewrote so as to be compatible with the \texttt{calc} package’s \texttt{\setlength} command (problem initially reported by Gary L. Gray and narrowed down by Martin Vaeth) \ldots \ldots \ldots \ldots 10
### Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

**Symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>@cons</td>
<td>81</td>
</tr>
<tr>
<td>@elt</td>
<td>7</td>
</tr>
<tr>
<td>\latex@warning@no@line</td>
<td>12, 27</td>
</tr>
<tr>
<td>\AtEndDocument</td>
<td>5</td>
</tr>
<tr>
<td>\eqboxwidth</td>
<td>86</td>
</tr>
<tr>
<td>\eq@must@rerunfalse</td>
<td>4</td>
</tr>
<tr>
<td>\eq@must@reruntrue</td>
<td>4, 65, 72</td>
</tr>
<tr>
<td>\eq@restorefont</td>
<td>30, 41</td>
</tr>
<tr>
<td>\eq@settowidth</td>
<td>38, 62</td>
</tr>
<tr>
<td>\eq@storefont</td>
<td>30, 40, 41</td>
</tr>
<tr>
<td>\eq@taglist</td>
<td>3, 24, 81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\eq@tempdim</td>
<td>2, 10, 69, 70, 74, 75, 83, 84</td>
</tr>
<tr>
<td>\eq@tempdimb</td>
<td>2, 9, 10, 62, 66, 67, 70, 71, 75, 76</td>
</tr>
<tr>
<td>\eq@parbox</td>
<td>46</td>
</tr>
<tr>
<td>\eq@parbox@i</td>
<td>46</td>
</tr>
<tr>
<td>\eq@parbox@ii</td>
<td>46</td>
</tr>
<tr>
<td>\eq@parbox@iii</td>
<td>49, 54, 58, 59, 61</td>
</tr>
<tr>
<td>\f@baselineskip</td>
<td>33</td>
</tr>
<tr>
<td>\f@encoding</td>
<td>32</td>
</tr>
<tr>
<td>\f@family</td>
<td>32</td>
</tr>
<tr>
<td>\f@series</td>
<td>32</td>
</tr>
<tr>
<td>\f@shape</td>
<td>32</td>
</tr>
<tr>
<td>\f@size</td>
<td>33</td>
</tr>
<tr>
<td>\fontsize</td>
<td>33</td>
</tr>
<tr>
<td>I</td>
<td>4, 26</td>
</tr>
<tr>
<td>\parbox</td>
<td>84</td>
</tr>
<tr>
<td>\RequirePackage</td>
<td>37</td>
</tr>
<tr>
<td>\selectfont</td>
<td>34</td>
</tr>
<tr>
<td>\settowidth</td>
<td>39</td>
</tr>
<tr>
<td>\usefont</td>
<td>32</td>
</tr>
<tr>
<td>W</td>
<td>14</td>
</tr>
</tbody>
</table>

---

12