LaTeX

Customizing LaTeX lists

Michel Goossens* , Frank Mittelbach† and Alexander Samarin‡

Abstract

In this article, which is part of the third chapter of our forthcoming book "The LaTeX Companion", we take a look at LaTeX's list environments. First the various parameters and commands controlling the standard LaTeX lists, `enumerate`, `itemize`, and `description`, are discussed. Then the general `list` environment is introduced and we tell you how to build custom layouts by varying the values of the parameters controlling this environment.

Introductory Remarks

As mentioned above, this article is part of our book "A LaTeX Companion", which will be published by Addison-Wesley in January 1994. The aim of this book is to be a real complement to Lamport's LaTeX user's guide and reference manual and to help you use this software even more productively. At the Aston TeX Conference last July, an upgrade of LaTeX was announced. All supplementary commands and functionality of this new version, LaTeX 2_ε, are discussed in this book. Below we give an overview of the contents of the various chapters whose titles should convey relatively clearly the subject area addressed in each case.

- Chapter 1: *Introduction*
  A short introduction to the LaTeX system.
- Chapter 2: *The Structure of a LaTeX Document*
  A discussion of generic and document-oriented markup.
- Chapter 3: *Basic Formatting Tools*
  LaTeX's basic typesetting commands.
- Chapter 4: *The Layout of the Page*
  An explanation of which tools are available to globally define the visual layout of the pages of a document by using pagestyles.
- Chapter 5: *Tabular Material*
  A tutorial on how to assemble material into columns and rows with the extended `tabular` and `array` environments, and their multipage equivalents—`supertabular` and `longtable`.
- Chapter 6: *Mastering Floats*
  A general treatment of floating material.
- Chapter 7: *Font Selection*
  A detailed discussion of LaTeX's New Font Selection Scheme (NFSS2) presenting its various user commands. It is shown how to add new fonts, both in math and text mode.
- Chapter 8: *Higher Mathematics*
  A review of the `amstex` package, which adds many powerful typesetting commands in the field of mathematics.
- Chapter 9: *LaTeX in a Multilingual Environment*
  A discussion of the problem of using LaTeX in a multi-language or non-English environment. The babel system and other language-specific packages are described.
- Chapter 10: *Portable Graphics in LaTeX*
  An overview of the field of device-independent graphics showing how the `epic`, `eepic` and other packages extend the possibilities of LaTeX's basic `picture` environment.
- Chapter 11: *Using PostScript*
  A description of how the PostScript page description language can turn LaTeX into a full blown graphics utility and how, via the NFSS, a user can choose a font from amongst hundreds of font families, available as PostScript Type 1 outlines.
- Chapter 12: *Index Generation*
  A discussion of the problems associated with preparing an index. The program *MakeIndex* is described in detail.
- Chapter 13: *Bibliography Generation*
  An overview of how LaTeX's companion program BibTeX tries to solve problems related to maintaining bibliographic data bases. Bibliographic styles are discussed and the format

*CN Division, CERN, CH1211 Genève 23, Switzerland, <goossens@cernvm.cern.ch>

†LaTeX3 Project Coordinator, Zedernweg 62, D55128 Mainz, Germany, <Mittelbach@mzdmza.zdv.Uni-Mainz.de>

‡ISO, 1, rue de Varembé, CH1211 Genève 20, Switzerland, <samarin@isocs.iso.ch>

of the BibTeX language is presented in detail to enable you to customize an existing style.

- Chapter 14: *LaTeX Package File Documentation Tools*
  A description of how to document the code of LaTeX packages using the doc package and its companion program docstrip.
- Appendix A: *A LaTeX Overview for Package and Class Writers*
  A review of how to handle and manipulate the basic LaTeX programming structures, and a discussion of the LaTeX 2_ε commands that describe the structure of package and class files. An overview of the extensions introduced by the `calc` package in the field of arithmetic operations, and the `ifthen` package for control constructs.
- Appendix B: *TeX Archive Sites*
  An explanation of how to get the files described in our book (and other LaTeX related software) from the various TeX archives.
- An annotated bibliography of twelve pages.
- An index of over thirty pages.

LaTeX is presently being rewritten under the coordination of one of the authors (Frank Mittelbach), Chris Rowley and Rainer Schöpf. This endeavor is called the LaTeX3 Project. A lot of the functionality described in the "Companion" as extensions to basic LaTeX will be available in that system: as part of the kernel, or in one of the extension packages. To help funding, half of the royalties from our book will go directly to the LaTeX3 Project. Therefore, when buying it, you not only obtain a handy, complete, and up-to-date reference to many important and useful packages available with LaTeX today, but you also actively contribute to making LaTeX more powerful and user-friendly in the future.

1 Modifying LaTeX Lists

Lists are a very general LaTeX construct and are used to build many of LaTeX's display-like environments. LaTeX's standard list environments: `enumerate`, `itemize`, and `description` are discussed in the next section, where we also show how they can be customized. The general list environment is discussed in section 2.

It is relatively easy to customize the three standard LaTeX list environments, and the three sections below will look at each of these environments in turn. Changes to the default definitions of these environments can be made globally by redefining certain list defining parameters in the document preamble, or they can be kept local.

1.1 Customizing an `enumerate` List

LaTeX's numbered list environment `enumerate` is characterized by the commands and representation forms shown in table 1 on the facing page. The first row shows the names of the counter used for numbering the four possible levels of the list. The second, and third rows are the commands giving the representation of the counters and their default definition in the standard LaTeX classes. Rows four, five, and six contain the commands, the default definition, and an example for the actual enumeration string printed by the list.

A reference to a numbered list element is constructed using the `\theenumi`, `\theenumii`, and other similar commands, prefixed by the command `\p@enumi`, `\p@enumii`, etc., respectively. The last three rows in the table show the command, its default definition, and an example for the representation of references. It is important that you are careful to take into account the definitions of both the representation and reference building commands to get the references correct.

We can now create several kinds of numbered description lists simply by applying what we have just learned.

Our first example redefines the first, and second level counters to use capital Roman digits and Latin characters. The visual representation should be the value of the counter followed by a dot. The default value of table 1 on the next page is used for the reference prefix command `\p@enumi`.

```
\makeatletter
\renewcommand{\theenumi}{\Roman{enumi}}
\renewcommand{\labelenumi}{\theenumi.}
\renewcommand{\theenumii}{\Alph{enumii}}
\renewcommand{\labelenumii}{\theenumii.}
\renewcommand{\p@enumii}{\theenumi.}
\makeatother
\begin{enumerate} \item \textbf{Introduction }
  \begin{enumerate}
    \item \textbf{Applications}  \newline
      Motivation for research and applications
      related to the subject.       \label{q1}
    \item \textbf{Organization}  \newline
      Explain organization of the report, what
      is included, and what is not. \label{q2}
```

	First level	Second level	Third level	Fourth level
counter	`enumi`	`enumii`	`enumiii`	`enumiv`
representation	`\theenumi`	`\theenumii`	`\theenumiii`	`\theenumiv`
default definition	`\arabic{enumi}`	`\alph{enumii}`	`\roman{enumiii}`	`\Alph{enumiv}`
label field	`\labelenumi`	`\labelenumii`	`\labelenumiii`	`\labelenumiv`
default form	`\theenumi.`	`(\theenumii)`	`\theenumiii.`	`\theenumiv.`
numbering example	1., 2.	(a), (b)	i., ii.	A., B.
prefix	`\p@enumi`	`\p@enumii`	`\p@enumiii`	`\p@enumiv`
default definition	`{}`	`\theenumi`	`\theenumi(\theenumii)`	`\p@enumiii\theenumiii`
reference example	1, 2	1a, 2b	1(a)i, 2(b)ii	1(a)iA, 2(b)iiB

Table 1: Commands controlling an `enumerate` list environment

```
  \end{enumerate}
  \item \textbf{Literature Survey}  \label{q3}
  \item \textbf{Proposed Research}  \label{q4}
\end{enumerate}
q1=\ref{q1} q2=\ref{q2} q3=\ref{q3} q4=\ref{q4}
```

I. **Introduction**
   A. **Applications**
      Motivation for research and applications related to the subject.
   B. **Organization**
      Explain organization of the report, what is included, and what is not.

II. **Literature Survey**

III. **Proposed Research**

q1=I.A q2=I.B q3=II q4=III

You can also decorate an `enumerate` field by adding something to the label field. In the example below, we have chosen the paragraph sign § as a prefix for each label of the first level list elements.

```
\renewcommand{\labelenumi}{\S\theenumi.}
\begin{enumerate}
\item text inside list, more text inside list,
      text inside list, \label{w1}
\item text inside list, more text inside list,
      text inside list, \label{w2}
\item text inside list, more text inside list,
      text inside list, more text inside list.
\end{enumerate}
w1=\ref{w1}  w2=\ref{w2}
```

§I. text inside list, more text inside list, text inside list,

§II. text inside list, more text inside list, text inside list,

§III. text inside list, more text inside list, text inside list, more text inside list.

w1=I w2=II

You might even want to select different markers for consecutive labels. For instance, in the following example, characters from the PostScript font Zapf-Dingbats are used. In this case there is no straightforward way for automatically making the `\ref` commands produce the correct references. You can, however, use the `dingautolist` environment defined in the package `pifont`, which is part of the PSNFSS system. Note also that we have used the `calc` package for doing the addition inside the `\setcounter` command.

```
\newcounter{local}\renewcommand{\labelenumi}
   {\setcounter{local}{25+\value{enumi}}%
    \ding{\value{local}}}
\begin{enumerate}
\item text inside list, more text inside list,
      text inside list, more text inside list;
\item text inside list, more text inside list,
      text inside list, more text inside list;
\item text inside list, more text inside list,
      text inside list, more text inside list.
\end{enumerate}
```

① text inside list, more text inside list, text inside list, more text inside list;

② text inside list, more text inside list, text inside list, more text inside list;

③ text inside list, more text inside list, text inside list, more text inside list.

Finally, for those who do not want to get involved in customizing these commands themselves, there exists a package `enumerate` (by David Carlisle), which redefines the `enumerate` environment with an optional argument specifying the style in which the counter has to be printed. This argument can contain any one of the tokens `A`, `a`, `I`, `i`, or `1` for typesetting the value of the counter using (respectively) the `\Alph`, `\alph`, `\Roman`, `\roman`, or `\arabic` styles.

Moreover, these letters can be surrounded by any strings involving any other TEX expressions; however the tokens A, a, I, i, or 1 must be specified inside a {} group if they are not to be taken literally.

The cross-reference commands `\label`, and `\ref` can be used as with the standard enumerate environment. Note, however, that with this style the `\ref` command only produces the chosen representation of the counter value—not the whole label. It prints the value in the same style as `\item`, as determined by the presence of one of the tokens A, a, I, i, or 1 in the optional argument.

```
\begin{enumerate}[EX i.]
\item text item one level one.
  More text item one level one    \label{LA}
\item text  item two level one.
   \begin{enumerate}[{example} a)]
   \item text item one level two.
    More text item one level two \label{LB}
   \item text  item two level two.
   \end{enumerate}
\end{enumerate}
\begin{enumerate}[{A}-1]
\item text item one level one for list two.
                                  \label{LC}
\item text item two level one for list two.
\end{enumerate}
This is how list entries are referenced:
`\ref{LA}', `\ref{LB}' and `\ref{LC}' or
more fully `EX~\ref{LA}.' and `A-\ref{LC}'.
```

This example generates the following output:

EX i. text item one level one. More text item one level one

EX ii. text item two level one.

example a) text item one level two. More text item one level two

example b) text item two level two.

A-1 text item one level one for list two.

A-2 text item two level one for list two.

This is how list entries are referenced: 'i', 'ii.a' and '1' or more fully 'EX i.' and 'A-1'.

1.2 Customizing an `itemize` List

For a simple unnumbered `itemize` list, the labels are defined by the commands shown in table 2 on the facing page.

To create a list with different labels, you can redefine the label-generating command. You can make that change local for one list, as in the example below, or you can make it global by putting the `\labelitemi` redefinition in the document preamble. The following simple list is a standard `itemize` list with a marker from the PostScript ZapfDingbats font for the first level label:

```
\newenvironment{MYitemize}{%
  \renewcommand{\labelitemi}{\ding{25}}%
          \begin{itemize}}{\end{itemize}}
\begin{MYitemize}
\item Text of the first item in the list.
\item Text of the first sentence in the second
      item of the list. And the second sentence.
\item This sentence in the text of the third
      item of the list.
\end{MYitemize}
```

And this is the result:

➜ Text of the first item in the list.

➜ Text of the first sentence in the second item of the list. And the second sentence.

➜ This sentence in the text of the third item of the list.

1.3 Customizing a `description` List

Using the `description` environment you can change the `\descriptionlabel` command that generates the label. In the following example the font for typesetting the labels is changed from bold to sans serif.

```
\renewcommand{\descriptionlabel}[1]%
         {\hspace{\labelsep}\textsf{#1}}
\begin{description}
\item[A.] text inside list,  text inside list,
      text inside list, more text inside list;
\item[B.] text inside list,  text inside list,
      text inside list, more text inside list;
\item[C.] text inside list,  text inside list,
      text inside list, more text inside list.
\end{description}
```

The above gives:

A. text inside list, text inside list, text inside list, more text inside list;

B. text inside list, text inside list, text inside list, more text inside list;

C. text inside list, text inside list, text inside list, more text inside list.

The standard LATEX classes set the starting point of the label box in a `description` environment `-\labelsep` to the left of the left margin of the enclosing environment, so that the `\descriptionlabel` command in the example

	First level	Second level	Third level	Fourth level
Commands	`\labelitemi`	`\labelitemii`	`\labelitemiii`	`\labelitemiv`
Definition	`$\bullet$`	`\bf --`	`$\ast$`	`$\cdot$`
Representation	•	–	∗	·

Table 2: Commands controlling an `itemize` list

above first adds a value of `\labelsep` to start the label aligned with the left margin.

2 Making Your Own Lists

Lists are generated by the generic environment `list`:

```
\begin{list}{default_label}{decls}
  item_list
\end{list}
```

The parameter *default_label* is the text to be used as a label when an `\item` command is issued without an optional argument. The parameter *decls* sets up the different geometrical parameters of the `list` environment (see Fig. 2 on the next page). That figure also shows the default values for those parameters. The parameters can all be redefined with the help of the `\setlength` or `\addtolength` commands.

Several LaTeX environments are defined with the help of `list` (for example `quote`, `quotation`, `center`, `flushleft`, and `flushright`). Note that these environments have only one item, and the `\item[]` command is specified in the environment definition.

As an example, we can consider the `quote` environment whose definition gives it the same left and right margins. The simple variant `Quote`, shown below, is identical to `quote` apart from the double quote symbols added around the text. Note the special precautions, which must be taken to eliminate undesirable white space in front (`\ignorespaces`) and following (`\unskip`) the text.

```
\newenvironment{Quote}% Definition of Quote
  {\begin{list}{}{%
      \setlength{\rightmargin}{\leftmargin}}
        \item[]``\ignorespaces}
  {\unskip''\end{list}}
\ldots\ text before.
\begin{Quote}
   Some quoted text, more quoted text.
   Some quoted text, more quoted text.
\end{Quote}
Text following \ldots
```

... text before.

> “Some quoted text, more quoted text. Some quoted text, more quoted text.”

Text following ...

General lists are often used for documenting computer commands or program functions. For instance, in the following examples `entry` and its variants are used. In each case the name of the topic being described is entered as the parameter of the `\item` command.

In the list below, the `\makelabel` command and the two geometrical parameters (`\labelwidth` and `\leftmargin`) are redefined.[1]

```
\renewcommand{\entrylabel}[1]{%
                  \mbox{\textsf{#1:}}\hfil}
\newenvironment{entry}%
  {\begin{list}{}{%
    \renewcommand{\makelabel}{\entrylabel}%
    \setlength{\labelwidth}{35pt}%
    \setlength{\leftmargin}%
                {\labelwidth+\labelsep}}}%
  {\end{list}}

\begin{entry}
\item[Description]
 Returns from a function.  If issued at
 top-level, the interpreter simply terminates,
 just as if end of input had been reached.
\item[Errors]   None.
\item[Return values]\mbox{}\\
           Any arguments in effect
           are passed back to the caller.
\end{entry}
```

Description: Returns from a function. If issued at top-level, the interpreter simply terminates, just as if end of input had been reached.

Errors: None.

[1] In this and some of the following examples, we have used the `calc` and `ifthen` packages.

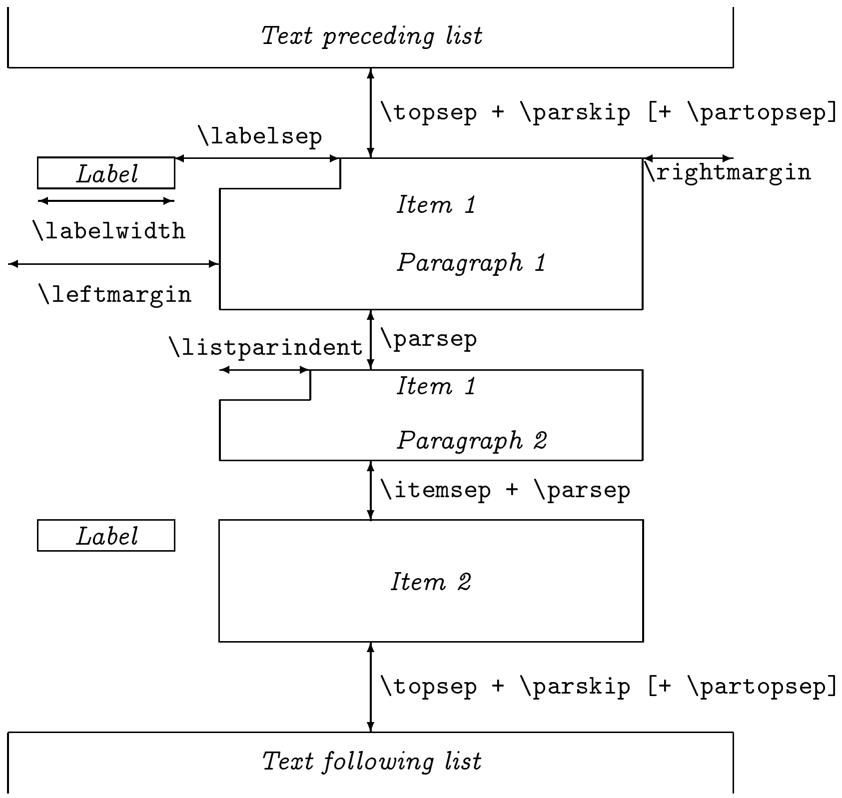

Figure 1: The structure of a general list

Vertical lengths

All the vertical spaces below are rubber lengths with a value depending on the type size and the level of the list.

`\topsep` Space between first item and preceding paragraph.

`\partopsep` Extra space added to `\topsep` when environment starts a new paragraph.

`\itemsep` Space between successive items.

`\parsep` Space between paragraphs within an item.

Horizontal lengths

`\leftmargin` space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative. Its value depends on the list level.

`\rightmargin` similar to `\leftmargin` but for the right margin. Its value is usually 0pt.

`\listparindent` extra indentation at beginning of every paragraph of a list except the one started by `\item`. Can be negative! Its value is usually 0pt.

`\itemindent` extra indentation added in front of an item label (not shown). Its value is usually 0pt.

`\labelwidth` the nominal width of the box containing the label. If the natural width of the label is ≤`\labelwidth`, then the label is typeset flush right inside a box of width `\labelwidth`. Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

`\labelsep` the space between the end of the label box and the text of the first item. Its default value is 0.5em.

Return values:
: Any arguments in effect are passed back to the caller.

This example shows a typical problem with description-like lists when the text in the label (*term*) is wider than the width of the label. Standard LaTeX lets the text of the term continue into the text of the *description* part. This is normally not desired, and to improve the visual appearance of the list we have started the description part on the next line. A new line was forced by putting an empty box on the same line, followed by '\\' command.

In the remaining part of this section various possibilities for controlling the width and mutual positioning of the term and description parts will be investigated. One method for accomplishing this is to change the width of the label. The environment is declared with an argument specifying the desired width of the label field (normally chosen to be the widest term entry). Note the redefinition of the `\makelabel` command where you specify how the label will be typeset. As this redefinition is put inside the definition of the `Ventry` environment, the argument placeholder character `#` must be escaped to `##` to signal LaTeX that you are referring to the argument of the `\makelabel` command, and not to the argument of the outer environment.

```
\newenvironment{Ventry}[1]%
  {\begin{list}{}{%
    \renewcommand{\makelabel}[1]{%
                        \textsf{##1:}\hfil}%
    \settowidth{\labelwidth}{\textsf{#1:}}%
    \setlength{\leftmargin}{%
                   \labelwidth+\labelsep}}}%
  {\end{list}}
```

Description: Returns from a function. If issued at top-level, the interpreter simply terminates, just as if end of input had been reached.

Errors: None.

Return values: Any arguments in effect are passed back to the caller.

However, several lists with varying widths for the label field on the same page might look typographically unacceptable. Evaluating the width of the term is another possibility. If it is wider than `\labelwidth`, an additional empty box is appended with the effect that the description part starts on a new line. This matches the conventional method for displaying options in UNIX manuals.

```
\newlength{\Mylen}
\newcommand{\Lentrylabel}[1]{%
 \settowidth{\Mylen}{\textsf{#1:}}%
 \ifthenelse{%
         \dimengreater{\Mylen}{\labelwidth}}%
     {\parbox[b]{\labelwidth}% term>labelwidth
        {\makebox[0pt][l]{\textsf{#1:}}%
                               \mbox{}\\}}%
     {\textsf{#1:}}% text<labelwidth
 \hfil\relax}
\newenvironment{Lentry}
   {\renewcommand{\entrylabel}%
     {\Lentrylabel}\begin{entry}}
   {\end{entry}}

\begin{Lentry}
\item[Description] Returns from a function.
  If issued at top-level, the
  interpreter simply terminates, just as if
  end of input had been reached.
\item[Errors]   None.
\item[Return values] Any arguments in effect
  are passed back to the caller.
\end{Lentry}
```

As the last line in this example shows, the `Lentry` environment is defined in terms of the `entry` environment. The label generating command `\entrylabel` is now replaced by the `\Lentrylabel` command. The latter first sets the length variable `\Mylen` equal to the width of the label. It then compares that length with `\labelwidth`. If the label is smaller than `\labelwidth`, then it is typeset on the same line as the description term, otherwise it is typeset in a zero width box with the material sticking out to the right as far as needed (forcing a new line) so that the description term starts one line lower.

Description:
: Returns from a function. If issued at top-level, the interpreter simply terminates, just as if end of input had been reached.

Errors: None.

Return values:
: Any arguments in effect are passed back to the caller.

Yet another possibility is to allow multiline labels. We, once more, use the `entry` environment as a basis, but this time the command `\Mentrylabel` replaces the `\entrylabel` command. The idea here is that large labels may be split over several lines. Certain precautions have to be taken to allow hyphenation of the first word in a paragraph, and therefore the \hspace{0pt} command is introduced in the definition. The material gets typeset inside a paragraph box of the correct width \labelwidth, which is then top aligned and left adjusted into a box that is itself placed inside a box with a depth of 1 em and no height. In this way, LaTeX does not realize that the material extends below the first line.

```
\newcommand{\Mentrylabel}[1]%
   {\raisebox{0pt}[1em][0pt]{%
      \makebox[\labelwidth][l]%
        {\parbox[t]{\labelwidth}{%
           \hspace{0pt}\textsf{#1:}}}}}
\newenvironment{Mentry}%
  {\renewcommand{\entrylabel}{%
    \Mentrylabel}\begin{entry}}%
  {\end{entry}}
\begin{Mentry}
\item[Description] Returns from a function.
  If issued at top-level, the interpreter
  simply terminates, just as if end of
  input had been reached.
\item[Errors]   None.
\item[Return\\values] Any arguments in effect
  are passed back to the caller.
\end{Mentry}
```

Descrip-tion: Returns from a function. If issued at top-level, the interpreter simply terminates, just as if end of input had been reached.

Errors: None.

Return values: Any arguments in effect are passed back to the caller.

An environment with an automatically incremented counter can be created by including a \usecounter command in the declaration of the list environment. This function is demonstrated with the Notes environment, which produces a sequence of notes. In this case, the first parameter of the list environment is used to provide the automatically generated text for the term part.

After declaring the notes counter, the default label of the Notes environment is declared to consist of the word NOTES in small caps, followed by the value of the notes counter using its representation as an arabic number followed by a dot.

```
\newcounter{notes}
\newenvironment{Notes}
   {\begin{list}{{\textsc{Note}}
      \arabic{notes}. }{\usecounter{notes}%
           \setlength{\labelsep}{0pt}%
           \setlength{\leftmargin}{0pt}%
           \setlength{\labelwidth}{0pt}%
           \setlength{\listparindent}{0pt}}}%
   {\end{list}}

\begin{Notes}
\item This is the text of the first note item.
      Some more text for the first note item.
\item This is the text of the second note item.
      Some more text for the second note item.
\end{Notes}
```

NOTE 1. This is the text of the first note item. Some more text for the first note item.

NOTE 2. This is the text of the second note item. Some more text for the second note item.

Nowa książka o LaTeX-u

Prezentowany wyżej tekst jest fragmentem książki *A LaTeX Companion*, która zgodnie z zapowiedziami autorów powinna zostać wydana przez Addison-Wesley w styczniu 1994 roku. Wstępna wersja książki była dystrybuowana przez autorów wśród uczestników tutorialu *Doing more with LaTeX 2.09* w czasie konferencji TeX-owej Aston '94. Możemy potwierdzić, że zamierzenia autorów o tym by *A LaTeX Companion* był uzupełnieniem dla książki Leslie Lamporta *LaTeX user's guide and reference manual* całkowicie się powiodły. W książce szczegółowo omawia się następujące tematy: LaTeX i PostScript, Nowy System Wyboru Fontów (NFSS), kilkaset znanych i nieznanych styli LaTeX-owych, użycie BibTeX-a i *MakeIndex*-a, pakiet AMS-TeX. Reasumując, nie jest to książka od której można rozpocząć naukę LaTeX-a, natomiast zaawansowany LaTeX-owiec (-ówka) może w niej znaleść wiele pożytecznych informacji i wskazówek.

Red.

Jeszcze jedna książka o LaTeX-u

W Wydawnictwie Uniwersytetu Wrocławskiego ukaże się drukiem w najbliższym czasie książka pt. *Wprowadzenie do systemu składu komputerowego LaTeX* autorstwa Jerzego Kucharczyka. Książka powstała na podstawie wykładów prowadzonych przez autora dla studentów w roku akademickim 1991/92. Objętość książki 15 arkuszy.

Red.